Algorithms and Performance measurements for MotifNetwork analysis programs

TR-09-02
Jeffrey L. Tilson, Gloria Rendon,
Eric Jakobsson

July 13, 2009

RENCI Technical Report Series
http://www.renci.org/techreports

Algorithms and Performance measurements for MotifNetwork analysis programs

Jeffrey L. Tilson[1], Gloria Rendon[2], and Eric Jakobsson[3]
1 Renaissance Computing Institute, University of North Carolina at Chapel Hill, U.S.A.
2 National Center for Supercomputing Applications, Urbana-Champaign, U.S.A.
3 Beckman Institute for Advanced Science and Technology, Department of Molecular and Integrative Physiology, University of Illinois at Urbana-Champaign, U.S.A.

Abstract

The MotifNetwork system is a high performance system for the fast scanning and interpretation of large numbers of proteins into their constituent domains. Once transformed into a domain dataset, several levels of analysis such as domain-domain and protein-protein co-location graphs are constructed. These basic data products form the beginning of a comprehensive environment for work in evolutionary processes with particular support for comparative analysis. MotifNetwork is based on a distributed architecture that has evolved into a reasonably secure system and is currently supporting researchers in drug target identification, ion-channels biophysics, functional orthologs, and socio-genomic processes.

To better support a broader research community, detailed analysis of the performance of several aspects of MotifNetwork are presented. Further, illustrative examples of using the data products in various matrix analyses are provided. Lastly, in conjunction with a recent submission to the BIOCOMP'09 conference [1], several remaining details on access, usage, and data archiving are summarized rendering fairly complete the technical details, architecture, and software components of the system as well as expected runtimes and results.[1]

1 Introduction

The genetic composition of an organism (its genotype) codes for the organism's physiology and behavior (its phenotype) in a complicated way. This relationship between genotype and phenotype depends on regulation at several levels of biomolecular function including transcription (production of the messenger RNA that codes for each protein), translation (production of the protein from the messenger RNA), and post-translational modification (adding other chemical groups to the protein, and protein-protein interactions). In the short term, the organism responds to pressures from the environment by altering expression patterns of different genes, so that the protein complement of the cell is different. Over a long (evolutionary) period of time the genome itself changes.

Dobzhansky famously state that nothing in biology makes sense except in the context of evolution. A corollary of this statement is that our ways of organizing biological information will not make sense unless we organize the data in ways that are consistent with pathways of evolution. Although Linneaus preceded Russell and Darwin, we now realize that the success of the Linnean classification system for organisms was due to the fact that the tree structure produced by mapping the classes followed the dominant pathway of evolution for organisms, especially eukaryotes; i.e., descent and small variations that over time that ultimately result in speciation of reproductively isolated populations. In eukaryotes, there is a vanishingly small amount of genetic recombination across species lines. The situation is somewhat more complicated for prokaryotes due to the relatively high incidence of lateral gene transfer among prokaryotes, but to a good first approximation the classification system and the resulting tree structure holds. We now see from the discussion above and the above-cited work that evolution of proteins is significantly different from evolution of organisms. A significant factor in the evolution of proteins, even in eukaryotes, is precisely the type of recombination of large building blocks from different families that practically never occurs at the species level.

Interpretation of evolutionary relationships of these building blocks depends on the ability to identify and classify them. It is natural to extend this concept to the organization of molecular biology information, and to create, for example, a gene ontology [2] based on the biological molecule. In this (canonical) view of living systems, proteins, and the corresponding genes, are taken to be the appropriate organizational units around which to build an understanding of the chemical basis of biology. The success of this approach is manifest in its myriad contributions to our current understanding of life and its evolutionary history.

---

[1] Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation

The weakness of this approach, however, emerges from considering the problem of classification, which is essential to organizing information. One can envisage two distinct approaches to protein family definition. One approach to protein family definitions is a statistical approach based on homology of sequences in complete genomes, and rigorous application of clustering theory. This approach, exemplified in [3], revealed 56,667 distinct protein families with a clear projection that the number of different protein families would continue to increase if the method were applied to additional complete genomes. This definition of families, while rigorously correct, is clearly of limited practical utility because of the enormous number of different families.

A second, more intuitive approach to defining a 'protein family' is exemplified in such projects as Transport Classification Data Base[2], the Molecule Pages project of the National Alliance for Cell Signaling[3], and in the data bases and analysis tools of the Human Genome Project[4]. These projects have great practical utility, because the number of families they present is reasonable in number and are intuitively clear. However they ultimately fail to provide logically clean classification systems, because many proteins are in fact hybrids among the protein families and classes, in the sense that proteins from different families may contain some of the same "domains" (relatively conserved parts of a protein, many of which parts have known functions, such as transport, catalysis, or regulation). This renders classification at the protein and gene level problematic, because of the continual need to deal with special cases that do not follow the classification system.

Working with an imperfect classification system has practical consequences. As one example from our laboratory found that a search for prokaryotic members of the Ach receptor channel family by a Blast search utilizing complete eukaryotic protein sequences found nothing of interest. Others had done a similar search, so that it had become conventional wisdom that this family was not represented in prokaryotes. However, when Asba Tasneem (a graduate student in our group) parsed the eukaryotic sequences into conserved domains and did pattern matching utilizing the various domains, there was revealed a group of prokaryotic members of the acetylcholine receptor channel family [4]. Subsequent experimental work confirmed the identity of this group [5]. Mao-Feng Ger, a student in our group, has taken a similar approach to identifying prokaryotic members of the glutamate receptor channel family with similar results; i.e., discovery of prokaryotic members of this family that are not revealed by PSI-BLAST search [6].

A potentially more useful taxonomy lies in classification by "domains" [7] many details of which are reviewed in [8]. Most present-day proteins have multiple functional domains [9] where a protein's "domain architecture" is comprised of the occurrence and position functional domains. Much of the evolutionary path of proteins appears to occur by rearrangement and addition of domains [10] which results in a model of protein evolution using a maximum parsimony analysis, the results of which suggest that the overall path of evolution has been towards greater complexity. Fusions, which cause an increase in the number of domains in proteins, have outnumbered fissions by 6 to 1). [11] provides the concept of a "domain distance" (degree of difference in domain architecture) as a measure of evolutionary distance between proteins.

2 MotifNetwork System

The MotifNetwork system has been developed to support the processing of large numbers of sequences into their domains, their co-location relationships and statistical properties. Our architecture, developments and plans go beyond simple domain relationships to creating a foundation for fast comprehensive systems biology approaches such as described in [12] in support of evolutionary correlation analysis. To render possible these large computations requires the incorporation of high throughput capabilities that exploit large grid-connected [13] supercomputers as found on Teragrid[14] and the OpenScienceGrid [15].

Deployment of a comprehensive domain-based analysis environment requires the preliminary construction of large datasets, tools to access and process data, and simple procedures for allowing users to benefit from the system. To this end, several workflows were created. These workflows are constructed to share several common technologies. More information has been published [16].

The first step to bringing new workflows to MotifNetwork is the selection of technologies. This is currently performed using the Taverna Workbench[17]. Several alternatives to Taverna exist including Pegasus [18], Kepler [19], and Triana [20], to name a few. There are also systems being developed that can utilize workflow descriptions from many systems [21]. We chose Taverna, however, because of the large number of included biological services and our substantial experience building Taverna-based workflows both for biological purposes and outside of the biological domain [22]. Though Taverna (v1.7.1) doesn't directly support accessing grid middleware as required by MotifNetwork, our technology choices for creating the biological services mitigates this problem.

The Taverna system is designed to use a centralized scheduling model. So for every enactment of a workflow, Taverna will manage the states of the services and the scheduling of the services accordingly. This also requires data to/from every service to

[2] http://www.tcdb.org/
[3] http://www.signaling-gateway.org/molecule/
[4] http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml

pass through the centralized server potentially resulting in memory constraints. We've gained experience in how to best decide what data should be returned directly through the server and which should be transferred as files. Performance results are discussed in the next section.

The Taverna workflows can be characterized as non-directed acyclic graphs. This is a powerful feature of Taverna (v1.7.1) and its underlying language, the Simple Conceptual Unified Flow Language (SCUFL), as it supports the use of implied iteration. This iteration allows workflows to be applied to arbitrarily long lists of data with one or more members (elements) and furthermore, allows the developer to specify customized ways to coordinate data arrays of differing cardinalities. Examples include matching an experiment number to every input sequence. Complex coupling arrangements, as found in the MotifNetwork, are difficult to implement and debug and generally requires staff with lots of experience. Taverna has basic Quality of Service (QoS) concepts that include the ability to retry a service that has failed. Though not sufficient for many of the failure modes identified during the MotifNetwork prototype developmental stages, this retry is useful.

The MotifNetwork environment leverages several community standard codes and data sets to create the final products. The most important ones include the stand-alone InterProScan (v4.4) [23] application and the associated Interpro (v19.0) [24] datasets, Cytoscape (2.4.1+) [25] (via creation of compatible files to facilitate analysis), and PSI-BLAST (v6.1) [26] application. The parallel analysis programs ScoreMatrix and WebMatrix were developed for MotifNetwork. These applications and databases must be installed onto the remote computer systems that they will be executed on.

Each of the Interpro databases has a particular approach to defining, identifying and characterizing domains/motifs; the approach could be based on sequence-based homology alone, or on experimental results only, or on a combination of both. For instance PROSITE uses regular expressions and profiles, PRINTS [27] uses PSSM (Position Specific Scoring Matrix)-based fingerprints, ProDom [28] uses automatic sequence clustering, and Pfam [29], SMART [30], TIGRFAMs[31], PIRSF[31] SUPERFAMILY[32], Gene3D [31], PROSITE [33], and PANTHER[31], all use hidden Markov models (HMMs). Given these differences, we generally use all databases in creating our data products to present final products that are unbiased. Subsequent filtering of the data is always possible.

2.1 MotifNetwork Workflows

Four fundamental workflows constitute the core of MotifNetwork. These workflows are named Wholegenome, Protein-Probe, ScoreMatrix, and WebMatrix. Each workflow is now described.

2.1.1. Wholegenome workflow

The Wholegenome workflow takes as input a large number of input sequences (~100,000) and processes them to identify domains and their inter-relationships. Procedurally it takes an input set and splits them into many disjoint subsets (100s) of sequences. These subtasks (chunks) form the set of independent jobs that may be executed concurrently by a version of InterProScan (IPRscan) installed on remote computers and accessed via grid-services. The resulting set of bipartite protein-domain partial graphs are combined (named the halfpair data) and further processed to generate several types of domain-domain and protein-protein pairwise results. Some details of the nature and values for several of the input values have been described [1].

A high level depiction of this workflow is collected in Fig 1. This workflow depiction uses boxes to indicate Taverna processors of various kinds. The blue colored boxes refer to I/O ports where data enters or exits the workflow/processor. Other boxes refer to computational steps. The details of all color combinations are available in the Taverna manual [34]. Many of the important custom processes will be discussed. They perform customized formatting and specialized services that perform the bulk of the computations. Connections between processes indicate dataflow. There are seven major steps to the Wholegenome workflow that are now summarized.

- InputSeqsFromiRODS: Input the name of the input file that contains the list of sequences to be processed
- SplitFastaList: This java shim processor takes the list of sequences and splits them into smaller sets of sequences This is a vital step as the degree of splitting dictates the subsequence workflow performance
- StoreIndividualSequencesLocally: Is a subworklow that stores the aforementioned smaller sequence sets (chunks) to uniquely named individual files
- InterProScan: This is a grid-service that uses implicit iteration to process the set of chunks. The results are a list of XML formatted output files

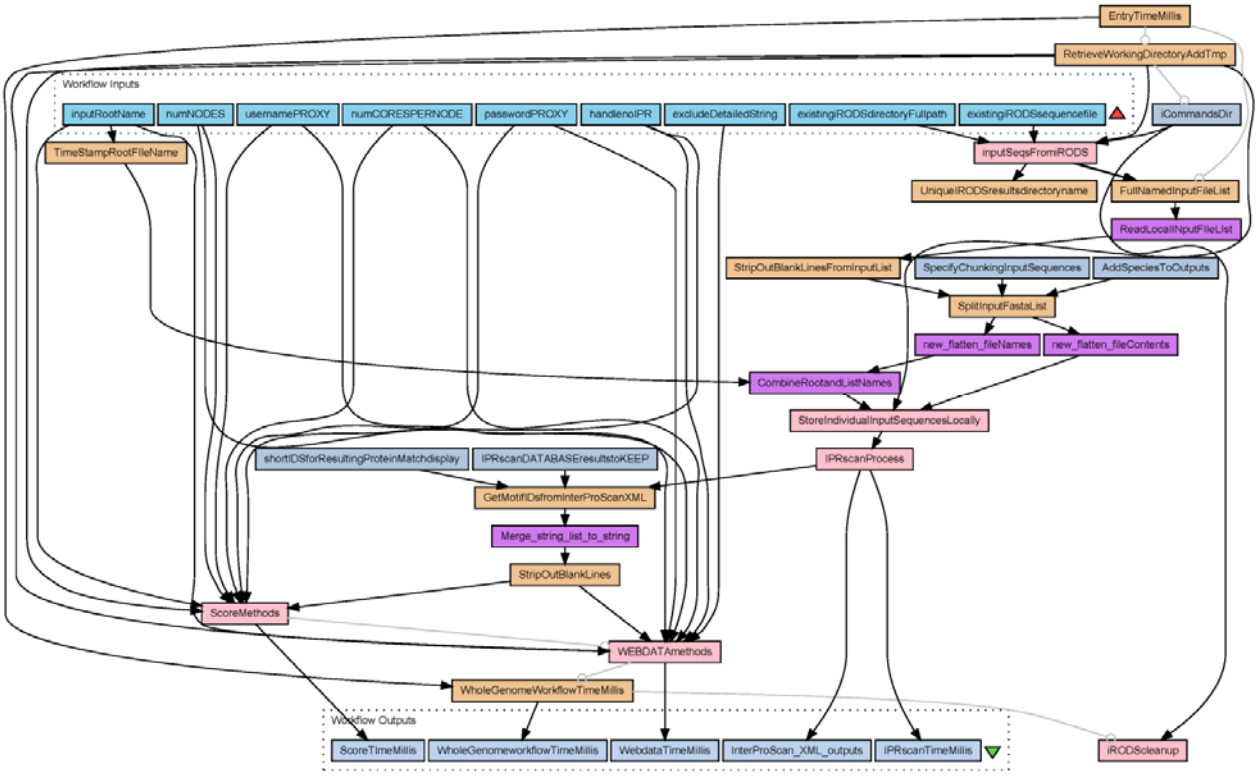

Fig. 1 Depiction of the Wholegenome workflow. Data flow generally is from top to bottom. Processing steps are indicated as colored boxes

- GetMotifIDsfomInterProScanXML: First processes each XML output file looking for valid IDs (per the user supplied input requirements) and lastly merges this set of results into a single large list. This list is referred to as the PseudoSIFs file or the file of halfpair
- WEBDATAmethods: Is a grid-service that takes as input the halfpair file and constructs several graphs based on proteins and domains. The underlying application is a parallel program
- ScoreMethods: Is a grid-service that takes as input the halfpair file and constructs two matrices that represent adjacency-like information for the biological system understudy. This application is also a parallel program

2.1.2 Protein-Probe workflow

The MotifNetwork workflow begins with one or more protein-type sequences and generates a complete domain-web for each. These initial sequences are first passed through a grid-service to a standard implementation of PSI-BLAST executing on a remote computer. The set of protein matches (based on a user selection criterion) including all unique matches are extracted from the PSI-BLAST results. A typical PSI-BLAST run is not particularly computationally expensive, but needs to be performed once for each input sequence. The set of PSI-BLAST identified targets are fetched from a local database (or alternatively from NCBI) and prepared for further processing. PSI-BLAST scans typically result in 1,000 sequences to be fetched and processed for each input probe.

Once fetched and additionally annotated, the domain analysis begins. The (remotely installed) IPRscan and application and Interpro databases are used to perform the basic scanning of the set of sequences. The halfpair results returned by IPRscan are assembled and for co-location information.

The Protein-Probe workflow pipeline is depicted in Fig 2. Select processors are now summarized.

- ProcessInputFileSequenceFormat: For this workflow only a single sequence is input
- PSIBLAST: A grid-service that invokes a remote instance of PSI-BLAST
- SplitFastaList: This java shim processor takes the list of sequences and splits them into smaller sets of sequences

- StoreIndividualSequencesLocally: Is a subworklow that stores the aforementioned smaller sequence sets (chunks) to uniquely named individual files
- InterProScan: This is a grid-service that uses implicit iteration to process the set of chunks. The results are a list of XML formatted output files
- GetMotifIDsfomInterProScanXML: First processes each XML output file looking for valid IDs (per the user supplied input requirements) and lastly merges this set of results into a single large list. This list is referred to as the PseudoSIFs file or the halfpair dataset
- WEBDATAmethods: Is a grid-service that takes as input the halfpairs file and constructs several graphs based on proteins and domains. The underlying application is a parallel program
- ScoreMethods: Is a grid-service that takes as input the halfpair file and constructs two matrices that represent adjacency-like information for the biological system understudy. This application is also a parallel program

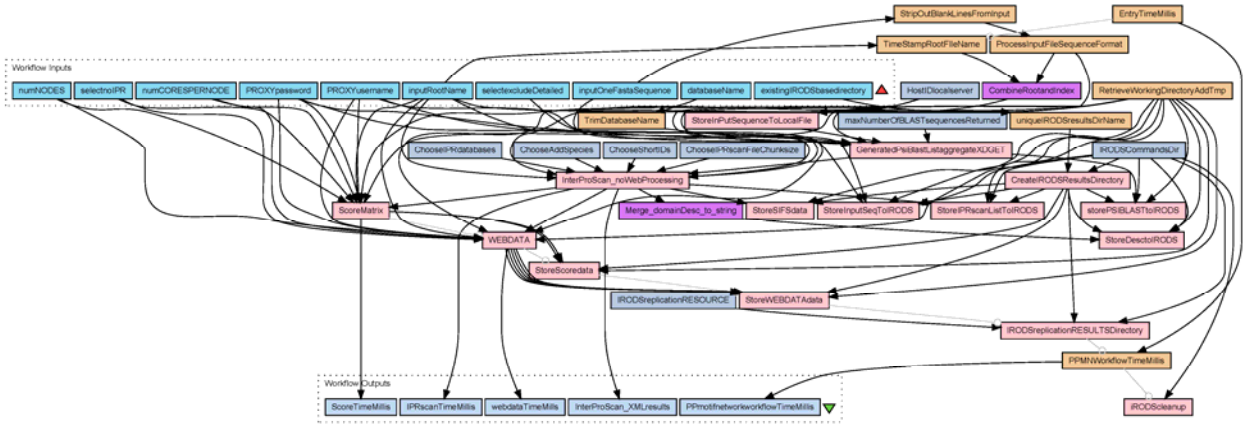

Fig. 2 Depiction of the Protein-Probe workflow. Data flow generally is from top to bottom. Processing steps are indicated as colored boxes

2.1.3 ScoreMatrix workflow

The ScoreMatrix subworkflow is displayed in Fig 3. This program processes the halfpair data to create an adjacency-like matrix that represent the proteinID vs domainID graph. The matrix elements are one or more likelyhood scores (eScore) and positions from IPRscan. Multiple scores are included when required. This data and filenames for these matrices are Protein_motif_DataBase_matrix.text and .Protein_motif_Position_matrix.text for the eScore and position matrices, respectively. Though not a stand-alone scientific workflow, the ScoreMatrix subworkflow is used by all workflows in MotifNetwork and thus warrants being summarized independently.

- ConstructValidFileandPathName: This processor take the halfpair data that is on-the-wire and stores it to a local file
- primitiveScoreMotifNetwork: This subworkflow transfers the halfpair data to a remote machine and launches a parallel application. The results are acquired and stored to new files for subsequent archive

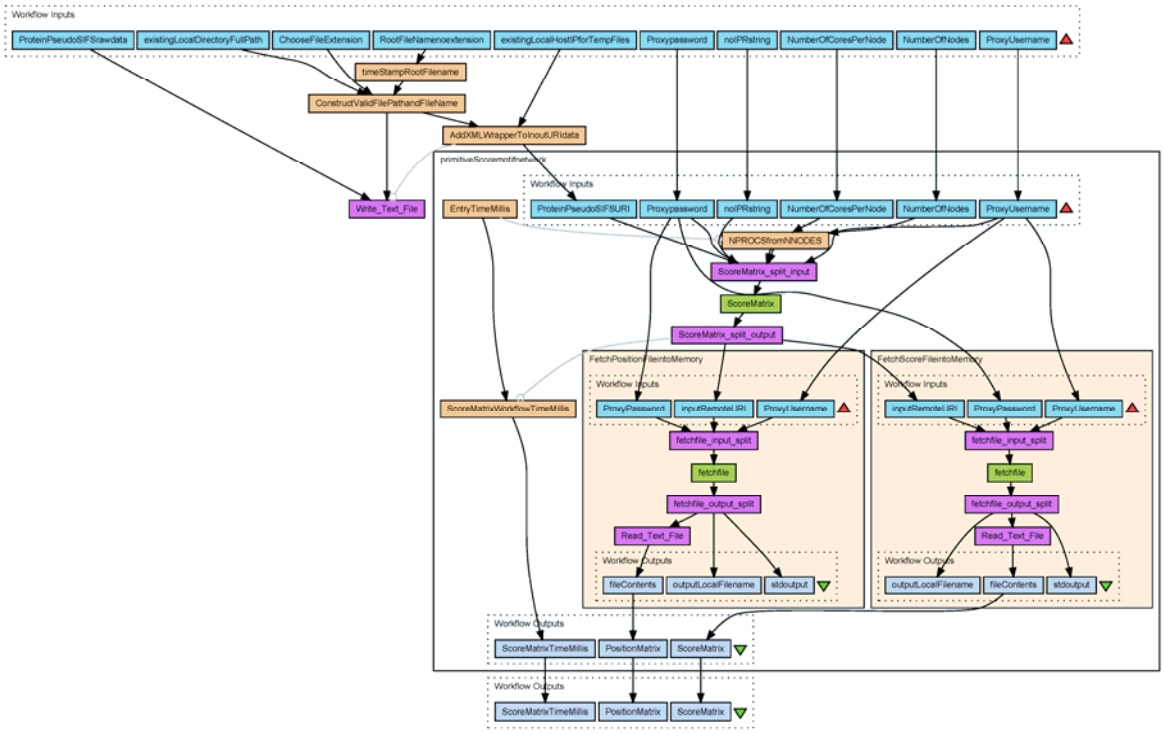

Fig. 3 Depiction of the ScoreMatrix workflow. In this layout, subworkflows have been explicitly exposed

- ScoreMatrix: A grid-service that performs the relevant computation on the halfpair data set.

2.1.4 WebMatrix workflow

The WebMatrix subworkflow is displayed in Fig 4. This process creates five data files useful for researchers. They are text files that may be processed by many applications. They are generally formatted, however, to be compatible with Cytoscape. Thus, the file extensions are used to denote their Cytoscape meaning.

1. MotifNetwork_WebDetailedData.sif . This data set is a sif formatted graph. Graph nodes indicate domainIDs, while vertices indicate the proteinID on which the 2 domains are found
2. MotifNetwork_WebSummaryData.sif is a sif formatted condensed graph . Nodes indicate domainIDs. Vertices are weighted quantities that indicate the number of proteinIDs the domain pair was found
3. MotifNetwork_MotifProtein.sif is a sif formatted bipartite graph the connects domainID nodes to proteinID nodes
4. MotifNetwork_radii.pvals is a Cytoscape attributes file that contains all domainIDs and the number of proteins thesew IDs were found. The file may be used by Cytoscape to generate node radii visual cues
5. MotifNetwork_InteractionStrength.pvals. This is an attributes file that specifies vertex weights for the WebSummaryData sif file

Lastly, a sixth file is made available with all workflow results. This MotifVizPropertyFile.props file is used by Cytoscape to apply the aforementioned attribute files to the sifs graphs.

The workflow processes for this subworkflow are nearly the same as for the ScoreMatrix workflow. Thus much of the relevant software has been reused.

- ConstructValidFileandPathName: This processor take the halfpair data that is on-the-wire and stores it to a local file.
- primitiveScoreMotifNetwork: This subworkflow transfers the halfpair local file to a remote machine and lunches a parallel application. The results are acquired and stored to new files for subsequent archive.
- WebdataMatrix isa grid-service that invokes a parallel program the performs the computation and create set of output files.

As with the ScoreMatrix subworkflow, the WebMatrix workflow is a subworkflow used by many MotifNetwork scientific workflows.

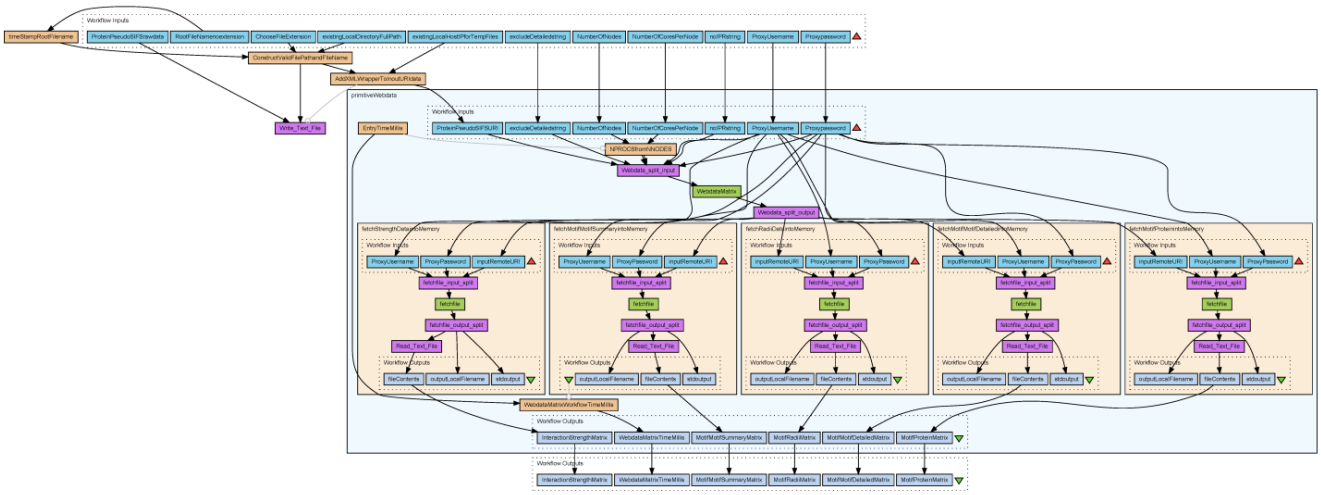

Fig. 4 Depiction of the WebMatrix workflow. In this layout, subworkflows have been explicitly exposed

2.2. Usage of MotifNetwork Workflows

To effectively use the MotifNetwork workflows in support of research requires understanding in some depth the input parameters used to launch these workflows. Details on how to actually gain access to the system has been reported [1]. Thus, the two most fundamental workflows that are most often used are now summarized. Several genomes are used to demonstrate the system. These were obtained from [35].

2.2.1. Wholegenome Workflow Usage

The Wholegenome workflows is basically an invocation of the InterProScan (IPRscan) application. While IPRscan can control chunking of input sequences and leverage a cluster environment for overall speed, our workflow is designed to perform much of this. Within our distributed grid environment it was best to launch non-parallel IPRscan jobs with pre-generated disjoint subsets of the input sequence list. We exploit the Taverna concurrent threads settings to perform IPRscan jobs concurrently. As a result, the workflow performs several data processing steps before and after the IPRscan process.

Running this workflow begins by specifying several input parameters. These are represented by the nine blue boxes representing the Workflow Inputs (Fig 1). These are required entries. The workflows acquire the indicated proxy from a server specified within the workflow. That can be overridden by the user as necessary.

- usernamePROXY. The username for a proxy that will be associated and delegated for all services. The proxy server for fetching this proxy is specified within the workflow but may be overridden by the user
- passwordPROXY. The associated myproxy passphrase for the proxy. Note this should not be the same as the passphrase used to create the actual proxy
- existingIRODSdirectoryFullpath. This input specifies an existing collection (directory) on the relevant iRODS collection that the user has the proper privileges and will contain the input sequences and the resulting file. There are specialized collections have been created to give MotifNetwork users have access and can create subcollections
- existingIRODSsequencefile. This specified the name of the file that contains 1 or more fasta formatted sequences for processing by the workflow. This file must reside in the iRODS collection specified by existingIRODSdirectoryFullpath
- inputRootName. This is a string value used in many places of the workflow. Names of temporary files are based on this as are names of related iRODS subcollections. A string that identifies the experiment is usually defined
- handlenoIPR is a switch for passing (or not) non-integrated (noIPR) Interpro results into the final data sets. The only value acted upon is “excludenoIPR’. All other values include noIPR results
- excludeDetailedString is a switch to exclude the calculation of the DetailedWeb network. This data object is fairly computationally expensive and not always needed. The value “excludeDetailed” excludes the computation otherwise it is included
- numNODES applies only to the ScoreMatrix (ScoreMethods) and WebMatrix (WEBDATAmethods) parallel computations

- numCORESPERNODE only apply to ScoreMatrix and WebMatrix computations. This is an integer value indicating the number of cores to use per node (specified by numNODES)

Additional parameters (constants) not part of the input parameters are visible in Fig 1 and are equivalent to the data in Table 2 found in [1]. The way in which these parameters influence the workflow is now described. First, the input sequences are fetched from iRODS and read into local memory. At that time a new subcollection is created (UniqueIRODS results directory) to store the final results. The set of sequences are then split into subsets (chunks) of at most "SpecifyChunkinginputSequences" sequences. This value is usually set to 20-30. At the same time, if the "AddSpeciesToOutputs" internal constant is set to 'y', any species information found in the sequence fasta titles is hyphenated and appended to the last ID of a compound ID. This cumbersome process results in subsequent IPRscan results carrying through the compound ID (sequenced-species) to final data products. Each chunk is stored into a unique file (StoreIndividual sequences locally) on the local machine running the workflow. The size of the collection of chunks dictates the total number of IPRscan jobs that will launched. These jobs are launched concurrently up to a user adjustable maximum concurrency.

Once the concurrent IPRscan jobs are completed, they are processed to fetch domain information and pass that to the ScoreMatrix and WebMatix subworkflows. All files are stored into the iRODS system in the location specified by the parameter existingIRODSdirectoryFullpath. The section on iRODS provides more details on files and storage.

2.2.2 Protein-Probe Workflow Usage

This workflow performs an InterProScan analysis on the set of proteins homologous to a starting protein (probe). The same ScoreMatrix and WebMatrix data structures are generated for each probe. So this workflow significantly reuses the software from the Wholegenome workflow. Details of the uses of this workflow have been reported [1]. Basically, the researcher chooses an input protein that is treated as a probe. This probe is generally well characterized with respect to function. The probe is processed and provided to a secure PSI-BLAST grid-service. Typically 7 iterations (rounds) are completed with each round reporting a set of homologous proteins from the chosen reference database. (currently nr or microbial). The complete list of unique IDs (and scores) are identified. For repeated IDs, the values from the latest iteration are kept. The actually sequences for the list of IDs are fetched and processed. Once these steps are completed, the workflow performs a Wholegenome operation on the data. Several research projects are leveraging the Protein-probe workflow for their research such as in [36].

Running this workflow begins by specifying several input parameters. These are represented by the ten blue I/O boxes representing the Workflow Inputs in Fig 2. These entries and their meanings are collected here.

- usernamePROXY. The username for a proxy that will be associated and delegated for all services. The proxy server for fetching this proxy is specified within the workflow but may be overridden by the user
- passwordPROXY. The associated myproxy passphrase for the proxy. Note this should not be the same as the passphrase used to create the actual proxy
- existingIRODSbasedirectory. This input specifies an existing collection (the base directory) in which all results will be placed
- inputOneFastaSequence. This input requires one properly formatted input sequence
- inputRootName. This is a string value used in many places of the workflow. Names of temporary files are based on this as are names of related iRODS subcollections. A string that identifies the experiment is usually defined
- handlenoIPR is a switch for passing (or not) non-integrated (noIPR) Interpro results into the final data set. The only value acted upon is "excludenoIPR'
- databaseName indicates which installed database PSI-BLAST should use. Currently only nr or microbial
- excludeDetailedString is a switch to exclude the calculation of the DetailedWeb network. This data object is fairly computationally expensive and not always needed. The value "excludeDetailed" excludes the computation otherwise it is included
- numNODES applies only to the ScoreMatrix (ScoreMethods) and WebMatrix (WEBDATAmethods) parallel computations
- numCORESPERNODE only apply to ScoreMatrix and WebMatrix computations. This is an integer value indicating the number of cores to use per node (specified by numNODES)

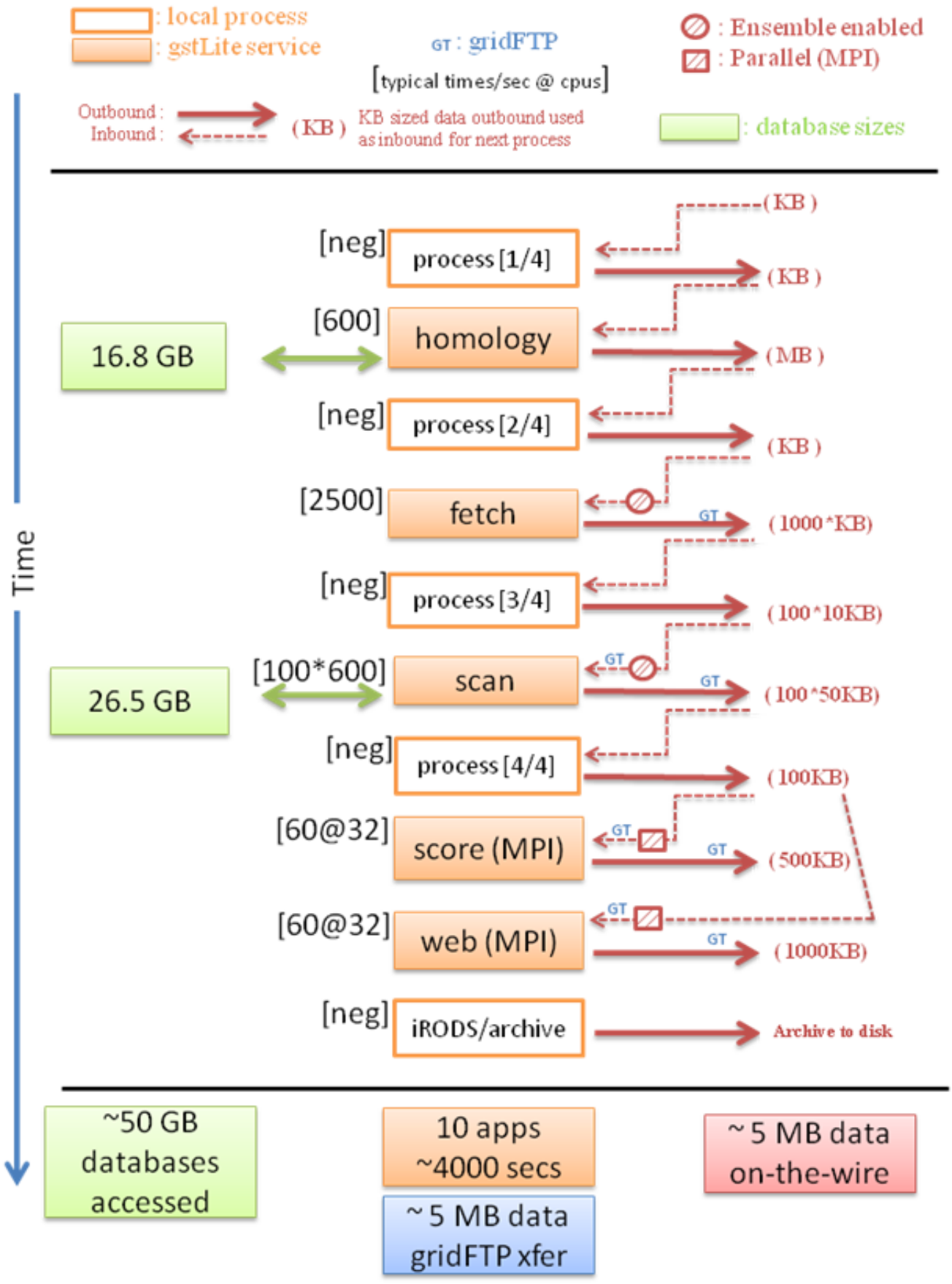

Fig. 5 Illustration showing typical networking, computation time, and database requirements for the Protein-probe workflow

2.3 Workflow performance characteristics

Characterization of the performance and behavior of a complex workflow is difficult. These issues are being actively addressed by the computer science community such as in [37]. In this section are specifications of the broad runtime characteristics attained through observations using typical data sets.

2.3.1 Protein-probe workflow

In Fig 5 is a representation of the Protein-probe workflow. Progress (time) through the workflow is indicated vertically from top to bottom. Three columns are apparent.

- The **center column** indicates basic processes (steps) in the workflow. Filled in boxes denote grid-services performing the named operation. Entries named 'process' are important shimming or data archiving operations. Associated with each process is an indicator of the typical time to execute the process. Neg = negligible. An entry such as [100*600] for the *scan* process indicates that approximately 100 concurrent jobs are invoked with each requiring 600 secs. An entry such as [60@32] for the *score* process indicates the job requires 60 sec on 32 processors.
- The **left column** is in indicator of the size of databases that are accessed by the indicated processors. This information is useful in indicating storage needs for partitulcar processors. As an example, the *homology* process (PSI-BLAST) access a set of databases (nr, microbial, etc.) that for MotifNetwork is of size 16.8 GB.
- The **right column** attempts to characterize complicated networking behavior. Most processes have an aggregate input (dashed) and output (solid) arrow. An added circle indicator to the input arrow indicates the likelyhood of concurrent transfers. These are usually clamped to a small number by the workflow. For these transfer types the number represents an aggregate value. The blue 'gt' indicator specifies the transfer is using the gridFTP protocol typically between grid-services. The values in parenthesis indicate the total volume of data transferred for that step. An entry such as (100*10K) indicates the transfer consists of 100 transfers each of size 10 KB.

At the bottom of the figure are cumulative values. For a common protein-probe workflow run, 5 MB of data are moved on-the-wire and ultimately sent to archival storage, ~50 GB of databases were accessed during the run, and the entire workflow required approximately 4,000 secs.

2.3.2. Wholegenome workflow

Fig 6 reports the analogous information for a typical run of the Wholegenome workflow. There are fewer processes and, in this case, only the Interpro databases (26.5 GB) are required. The results indicate a runtime of 21,000 sec and approximately 2GB of data moved on-the-wire and eventually archived.

2.4 Miscellaneous Components of MotifNetwork

The MotifNetwork is a complex distributed system. To effectively use it requires knowing something about many aspects of it. The following information enhances information from [1,16].

2.4.1 MotifNetwork gstLite services

The benefits of using workflow technologies and distributed and parallel processing available through a computational-grid are realizable only when grid-services of important biological applications can be easily created and invoked by the workflow. This section collects information on the newest version of the gstLite software used by the MotifNetwork project. See also [1].

The MotifNetwork system is a highly layered structure using different kinds of services and applications to perform the computations. This layering basically orders as users: workflows: services: applications. MotifNetwork developments require addressing issues at all layers. Particularly important, however, are the creation of the services. These services must manage the remote launching of jobs and associated file transfer operations. The Generic Service Toolkit (gstLite)[38,39] was selected to create the services. gstLite provides the ability to write a service that will invoke applications resident on a remote machine using grid-middleware while providing a simple web-services (wsdl) interface that Taverna can support. GST uses grid-services [40] to manage the movement of data and files between the service computer and the application computer and launches the application using standard batch systems such as PBS. There are no restrictions to using parallel programs such as those utilizing the MPI standard [41] for the application. gstLite creates grid services as Globus grid-services using the Java CoG [42] to interface with Globus GT4 [43]. The Grid Resource Allocation and Management (GRAM) [44] and GridFTP [45] and are used the handle our

remote launches. Security is handed using the Grid Security Infrastructure (GSI) [46]. The MyProxy [47] repository is used for managing credentials.

The process of creating a biological service is referred to as "wrapping" the application. The specific details of wrapping an application are extensive, tedious and described in the gstLite literature. However, overall they are not difficult to create for many kind of applications used by researchers. In summary, however, we first begin with the creation of three description files (xml) files. These are named the ServiceMap, ApplicationDescription, and HostDescription files, respectively. Some details of extensions to include security aspects and interoperability with Taverna. [1] are available.

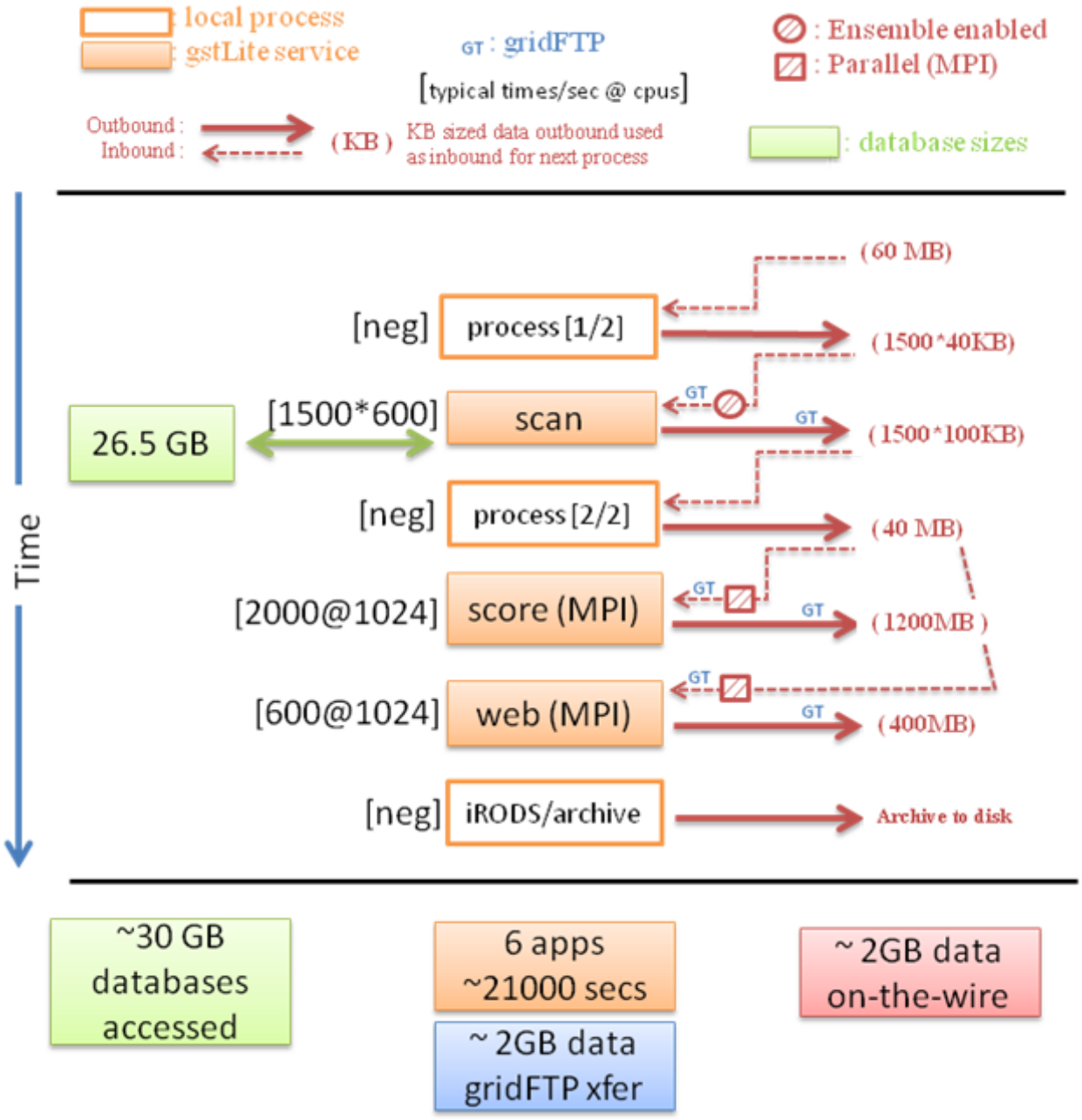

Fig. 6 Illustration showing the typical networking, computation time, and database requirements for the Wholegenome workflow

2.4.1.1 ServiceMap description

The ServiceMap file is an application specific configuration file which describes the interface between the service and the application itself. For the Wholegenome interpro service, this includes the input sequence ID, the minimum open reading frame size (default to 100), and the output format (specified as xml). These inputs correspond closely to those found at the EMBL-EBI community website [48]. One of the benefits of using gstLite is the ability to specify a generalized URI input/output data type. It is through this URI specification that gridFTP transfers can be requested between the GST Services and the computational resources without the need for the service to expose the underlying grid technologies to the user. The output for IPRscan is passed back to the service output port becoming available to the upstream workflow process.

This specification describes the input and output data types for our service. In this case we exposed three input objects that require specification at runtime. Two are simple string inputs corresponding to the *trlen* and *format* specifiers. The third entity is a Uniform Resource Identifier (URI) type object that corresponds to the sequence input filename. The URI type indicates to gstLite that the gridFTP protocol should be used to transfer the files. The MotifNetwork user never interacts with this underlying process.

```
<!-- ServiceMap file -->

<ServiceMap xmlns="http://www.renci.org/namespaces/2008/09/gstLite"
        xmlns:renci="http://www.renci.org">

  <service>
    <serviceName targetNamespace="http://www.renci.org">iprscan-service</serviceName>
      <serviceDescription>IPRscan 4.4</serviceDescription>
  </service>

  <portType>
  <method>
    <methodName>InterProScan</methodName>

    <methodDescription>Access to the InterProScan application</methodDescription>
    <application paramValuesOnly="false">
      <applicationName targetNamespace="http://www.renci.org">iprscan</applicationName>
        <description>iprscan application</description>
      </application>

    <inputParameter>
       <parameterName>-i</parameterName>
       <parameterDescription>Input file containing protein sequence</parameterDescription>
       <parameterType>URI</parameterType>
    </inputParameter>

    <inputParameter>
       <parameterName>-trlen</parameterName>
       <parameterDescription>Transcript length threshold (20-150).</parameterDescription>
       <parameterType>Integer</parameterType>
    </inputParameter>

    <inputParameter>
       <parameterName>-format</parameterName>
       <parameterDescription>Output results format</parameterDescription>
       <parameterType>String</parameterType>
    </inputParameter>

    <outputParameter>
       <parameterName>OutputParam1</parameterName>
       <parameterDescription>An output parameter</parameterDescription>
       <parameterType>StdOut</parameterType>
    </outputParameter>
  </method>

  </portType>
</ServiceMap>
```

2.4.1.2 Application Description file

The ApplicationDescription file specifies to the gstLite service "where" on the remote machines the application resides. Though typically thought of as pointing directly to the application of interest, MotifNetwork deployments typically point to a remote "script" that launches the application. This additional abstraction has been helpful in ensuring that application environments are properly set and that some basic kind of errors get managed cleanly.

The application description file for the IPRscan service is displayed. Specified are the maximum walltime (mins) for running the job, a specific computer to run the jobs (khawk), a location of the executable program on the remote computer (iprscan_wrapper.sh-*the script*), and the basename for a working directory. For an actual computation, gstLite creates a subdirectory under the basename (tmpDir) to perform all computations. This subdirectory name is time-stamped to minimize name clashing. An example of such a real name directory name is
/tmp/motifnetwork/COMMUNITY-SCR/SCR iprscan_Tue_Apr_21_10_27_01_EDT_20093679_Dir_remote

```
<!-- Simple application description document for the TestApp application on khawk.renci.org -->
<ApplicationDescription xmlns="http://www.renci.org/namespaces/2008/09/gstLite"
        xmlns:renci="http://www.renci.org">

   <applicationName targetNamespace="http://www.renci.org">iprscanid</applicationName>
   <maxWallTime>2000</maxWallTime>
   <deploymentDescription>
      <hostName>khawk.renci.org</hostName>
      <executable>/tmp/motifnetwork/APPS/iprscan/Service/iprscan_wrapper.sh</executable>
      <tmpDir>/tmp/motifnetwork/COMMUNITY-SCR/SCR</tmpDir>
   </deploymentDescription>
</ApplicationDescription>
```

2.4.1.3 Host description file

The **HostDescription** file specifies "how" to launch jobs to the remote machine. All of the important MotifNetwork workflow applications are launched using Globus (GRAM) usually to a remote job manager such as openPBS sometimes to a Fork manager. This is a fairly standard HPC computing environment. The name of the computer is supplied (khawk.renci.org) as are the details on how the application of interest should be launched. The value of tmpDir in this specification is irrelevent as we also specified it in the Application Descriptiop file which takes precedence.

```
<!-- A simple host description document for khawk.renci.org -->
<HostDescription xmlns="http://www.renci.org/namespaces/2008/09/gstLite"
        xmlns:renci="http://www.renci.org">
   <documentInfo>
      <documentName targetNamespace="http://www.renci.org">khawk.renci.org</documentName>
   </documentInfo>
   <!-- Name of the host -->
   <hostName>khawk.renci.org</hostName>
   <hostConfiguration>
      <tmpDir>/tmp</tmpDir>
            <gram>
                  <epr>khawk.renci.org/jobmanager-pbs</epr>
                  <type>PBS</type>
            </gram>
   </hostConfiguration>
</HostDescription>
```

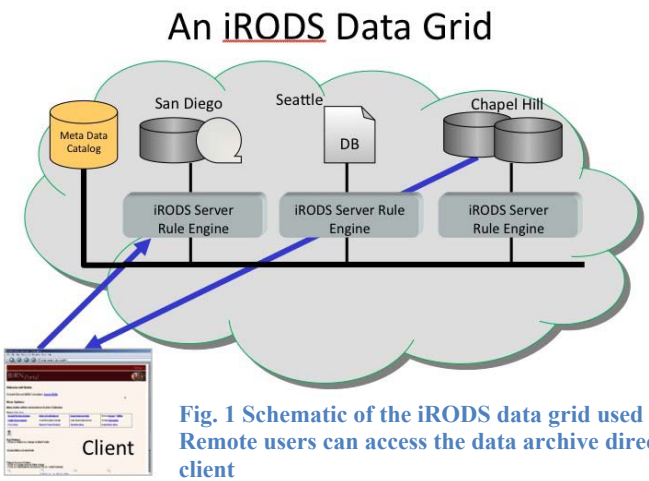

Fig. 1 Schematic of the iRODS data grid used by MotifNetwork. Remote users can access the data archive directly through the web client

2.4.2 iRODS: MotifNetwork data storage and archive

MotifNetwork utilizes an iRODS collection [49] to organize results at the completion of a run. iRODS is a sophisticated data-grid system that can account for multiple kinds of hardware and organizational (metadata) constructs, has a simplified procedure for building system client for remote access, and data-level operations such as replication onto a tape archival system. This is accomplished by using a client/server architecture that uses distributed storage and compute resources. A database system is used for maintaining the attributes and states of data.

In Fig 7 is illustrated a representation of the data-grid used by the MotifNetwork system. Instead of using a web client as in the figure, MotifNetwork workflows access the system directly.

Several ways to access an iRODS collection are possible. These include, programming to the iRODS API, the use of microservices, and the direct use of the command line 'i' commands. MotifNetwork workflows currently leverages the Java Runtime methods to interface with the iRODS 'i' commands. Generally the iRODS client software is installed on the server running the Taverna Remote Execution Service (RES)[See 1] and on every remote system used by collaborators wishing independent access.

To facilitate incorporation of iRODS capability into MotifNetwork workflows, several subworkflows have been created. These generally emulate the suite of 'i' commands. They can be easily imported to workflows as required. Below are collected many of these subworkflows and a brief description of their function. The associated iRODS 'i' commands are included for comparison. In the following the parameter: *iRODs-collection* indicates the specification of a full pathname.

Subworkflow name and description	**iRODs equivalent command-line operation**
iRODSiget: This procedure reads the contents of the specified iRODS file into memory as a large string. A temporary file is created in a user-specified location	*iget iRODs-collection collectionFilename localDirectory localFilename*
iRODSiput: This subworkflow copies data in local (workflow) memory into a user specified file in the iRODS collection. A temporary file is created in a user-specified location.	iput *localDirectory localFilename iRODs-collection collectionFilename*
iRODSmkdir: This subworkflow simply creates a new collection	*imkdir iRODs-collection*
iRODcd: This subworkflow changes the default working directory.	icd *iRODs-collection*
iRODls: This subworkflow lists the contents of the current working directory. This is used to provide information to MotifNetwork workflows.	ils *iRODs-collection*
iRODrepl: This subworkflow caused the user specified collection to be replicated onto MotifNetwork mass storage. Generally this occurs at the completing of a large calculation. *Resource* refers to the name of the mass storage device	*irepl –rV –Rresource iRODs-collection*
wireToIRODS: This subworkflow is similar to iRODiput. Here data stored in memory (on-the-wire) is assembled into the proper form, and stored to a user specified file in the iRODS collection	iput *localDirectory localFilename iRODs-collection collectionFilename*
dataListToIRODS: This subworkflow performs several steps but is based on wireToIRODS, iRODSmkdir,and iRODSicd. Here a user specified collection is created. Then *wireToIRODS* is applied to a list of data objects in memory. These might be, for example, a list of InterProScan results returned by a large ensemble run. Each member of the list is stored to a unique file into the specified collection. File uniqueness is maintained by constructing a set of file names from a list index and a user supplied rootname	imkdir iRODs-collection icd iRODs-collection iput localDirectory localFilename iRODs-collection collectionFilename
iRODSiexit: This subworkflow simply attempts to cleanly exit from the iRODS system.	iexit

When MotifNetwork workflows are launched, one of the required common input parameters is the specification of an existing iRODS collection. As data are created by the workflows, they get stored onto the specified iRODS collection. In addition, two new collections (subdirectories) are created. These are useful in organizing the data because of the large number of files that stored. The first subcollection contains the list of xml-formatted InterProScan results. A typical genome computation can yield 1,000s of these files. The second collection contains the remaining results files.

Assembled is a listing of a real collection pertaining to a single genome run for the experiment named Mus_musculus. The input file is named Mus_musculus.faa and is read by the workflow from this collection. The contents of the example collections are displayed in red type. The Mus_musculus input contains 34,966 input sequences that are split into 1,166 chunks.

Consider the base directory from which the Mus_musculus input can be found.

/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST
containing the input file Mus_musculus.faa

The workflow creates a new collection (subdirectory) with a name based on the user specified base directory (/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST) and the string *RESULTS*, an experiment name (for this example that value is Mus_musculus.faa.postprocessed.fasta.V19.0) and time stamp generated at runtime (1238689236273) to ensure uniqueness. The actual name becomes:

/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST/RESULTS-
Mus_musculus.faa.postprocessed.fasta.V19.0-1238689236273

The contents of the *RESULTS* directory include the 13 relevant output files and another collection named iprscan_xml-1238836212336. This collection is also created by the workflow (using the *dataListToIRODS* subworkflow) to retain the 1,166 IPRscan output files. The collection name is based on a workflow-supplied basename and a time-stamp.

/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST/RESULTS-
Mus_musculus.faa.postprocessed.fasta.V19.0-1238689236273:
MotifNetwork_InteractionStrength.pvals
MotifNetwork_MotifProtein.text
MotifNetwork_WebDetailedData.text
MotifNetwork_WebSummaryData.text
MotifNetwork_radii.pvals
ProteinVSMotifs_pseudo_SIF_format.text
Protein_motif_DataBase_matrix.text
Protein_motif_Position_matrix.text
domainDescriptions.txt
runPBS.e1030911
runPBS.o1030911
runWeb.e1030913
runWeb.o1030913
C-/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST/RESULTS-
Mus_musculus.faa.postprocessed.fasta.V19.0-1238689236273/iprscan_xml-1238836212336

A partial list of the 1,166 file that are contained within the iprscan_xml-1238836212336 collection is

/nara-renci-irods/home/motifnetwork/GENOMES/LARGELIST/RESULTS-
Mus_musculus.faa.postprocessed.fasta.V19.0-1238689236273/iprscan_xml-1238836212336:
iprscan_xml-0.text
iprscan_xml-1.text
iprscan_xml-10.text
iprscan_xml-100.text
iprscan_xml-1000.text
iprscan_xml-1001.text
iprscan_xml-1002.text
(snipped)

This collection hierarchy structure is consistent across computations to facilitate users access to the large amounts of multi-genome data.

Once an experiment is completed and the data are stored remote users and collaborators may access to the data via the web client (Fig 7). Each user is provided an iRODS account which permits them to log into the MotifNetwork iRODS collections. Once there, they may peruse and download data that has been stored (given suitable permissions). By default, MotifNetwork stores data into collections that permit access to all users. Under select conditions, a workflow user can choose a more discriminating collection for their data.

2.4.3 Security Additional Security issue: Proxy certificates

Actually performing computations on the MotifNetwork system requires the availability of a valid grid proxy stored into a myproxy server that is accessible to the workflow. Many of the computationally expensive procedures have these additional authorization steps to ensure productive use of resources. This added security is based on the concept of a proxy. The proxy is a short-term, self-signed certificate that can be delegated to remote grid-services without the need to supply a password. The preferred way is for a user to create their own proxy certificate. This requires an X.509 grid-certificate that has been provided by

the MotifNetwork organization to each user. To actually create the proxy, the user needs to login to a grid-hosting environment provided by the MotifNetwork team and create the proxy and load the proxy into a specified proxy server. When a running workflow attempts to invoke a gstLite service, the service attempts to checkout a valid proxy using user supplied username/password pairs. If this is unsuccessful, the service will not be invoked. This method of creating a proxy provides full access to all MotifNetwork resources.

There is an alternative approach for beginning users needing only limited access. MotifNetwork provides a community account username/password that can be used by the workflows. Constraints have been applied to the use of these community accounts, however, to encourage users to get bonefide X.509 credentials. The constraints include a limitations on the degree of parallelism for the analysis programs and limitations on the ability to execute jobs concurrently. These limits permit computation on a small set of sequences but will be inadequate for whole-genome based analysis.

To actually create a user proxy requires first logging onto the RENCI machine as directed by MotifNetwork staff. Then one simply performs the following operation. Substitute the myproxy server name for server.name and a numerical value for the total estimated number of hours your workflow will run. In the example below a value of 10 is specified.

prompt> myproxy-init –s server.name –c 10

The results of invoking this command follows. The GRID passphrase is that chosen by the user when creating their original certificate request. The MyProxy pass phrase is a temporary passphrase that is used for the workflow input parameters.

Your identity: /O=RENCI/OU=Globus/OU=motifnetwork.org/CN=MotifUser
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Proxy Verify OK
Your proxy is valid until: Sat Apr 25 13:42:00 2009
Enter MyProxy pass phrase:
Verifying - Enter MyProxy pass phrase:
A proxy valid for 24 hours (1.0 days) for user MotifUser now exists on motifnetwork.org.

As a final note, the input entries for Taverna workflows get encrypted so as not to be stored into MotifNetwork server logs in a readable manner.

3 Performance

A core premise of our project is that progress in understanding function and evolution of proteins, and application of that understanding to biomedicine, would be facilitated by a high-throughput computing environment that can do full comprehensive domain and motif analysis of protein sequences rapidly enough to keep up with the increasing number of available genomes. Some preliminary performance results have been reported [50].

Generally, we find that with the application of a sufficient number of processor cores a full genome domain-transformation with analysis can be completed in less than 1.5 days with most in less than one day. Assuming that the theoretical compute rates for a typical server (Dell 5150) is 21.3 GFlops/s (Billions of floating point operations per second) per node (5.32 GFlops/s per core) then access to approximately 0.7 TFlops/s should result in processing one genome per day. We still see between 1000 and 2000 sequences processed per hour. This consistency suggests that we will be able to make reasonable predictions for computing resources for future genome calculations.

Table 1 Runtime in hours (hr) for the Wholegenome workflow on a maximum of 128 cores

Genome	input sequences	Runtime(hrs)	Sequences/hr
Saccharomyces cerevisiae	6,714	6.4	1049
Anopheles gambiae	13,133	6.0	2189
Mus musculus	25,707	13.5	1904
Apis mellifera (OGS + ab initio)	34,966	21.3	1642
Bovine glean5	26,835	31.2	860
Homo sapiens	70,509	32.7	2156
Escherichia coli W3110	4,226	1.7	2485

3.1 Ensemble Performance

Here we report early performance measurements for computationally significant parts of the system. We've identified the need for both capability and capacity computing concepts for high overall performance. Capacity (ensemble) computing pertains to managing hundreds of independent jobs that can execute concurrently. Generally, these jobs have no data interdependencies except for the need to search through databases. MotifNetwork uses ensemble techniques to perform fast domain scanning. Parallel-processing jobs are used to process the produced large domain lists (usually ~50,000 proteins) to calculate frequencies, and co-location results. Explicit parallel processing is used for these jobs.

Overall performance of the Protein-Probe workflow is limited by the non parallel PSI-BLAST as previously reported [16] and is not further analyzed in this report. The Wholegtenome workflow is designed to perform large numbers of concurrent domain scans. The level of concurrency is specified within Taverna using the "concurrent threads" setting.

Fig. 8 reports the runtime for the ensemble step as a function of the maximum possible number of concurrent jobs (degree of concurrency). The input genome is from E.coli (W3110) which contained 4,226 non-duplicate input sequences and yields 3,824 proteins that contain an identified domain and a total of 3,861 domains (Interpro v16.1) The calculations were orchestrated using the constants as collected in Table 2. The relatively small genome size (4,226 seqs) is agglomerated into 1,057 chunks. Each chunk processes at most 4 sequences. In a related publication in [1] are reported ensemble performance values for larger chunk sizes. The larger chunk sizes preclude using hogh degrees of concurrency owing to the starvation of work for the processors.

Table 2 Protein-Probe and Wholegenome workflows. Adjustable constants used for performance measurments

Constant Name	Setting
ChooseShortIDs	yes
ChooseIPRscanFileChunkSize	4
ChooseAddSpecies	f

The computer used for the measurement was not dedicated and so it was occasionally processing jobs for other users. Generally, though, it seemed that the workflow wasn't waiting for compute resources. Moreover, these walltimes include bottleneck steps such as the moving of results to archival storage and the replication of that collection onto tape backup Nevertheless, the overall wall time scales well with the degree of concurrency. The workflow decreases from 22.4 hours to 3.3 hr. Extrapolation back to a single threaded computation implies a runtime of 89.6 hr. A quantitative measure of scaling can be computed using Eqn 1. Here, is the relative concurrency effectiveness; CE(%), as a function of degree of concurrency. These CE(%) values are plotted in Fig X. Perfect efficiency would be 100% for all degrees (d). The WholeGenome workflow drops to approximately 60% at 48-way concurrent. This is a good result given the broadness of the included operations captured in the wall time measurements such as data archive and replication, file chunking processes and sequence validation, the transfer of 1,057 files to/from the computing resource, and the submission of jobs to a remote job manager (openPBS) on a computer not dedicated to our benchmarking.

$$CE(\%) = \frac{100 \times t(d_o) \times d_o}{t(d) \times d} \qquad 1)$$

$$t(d) = workflow\ runtime\ for\ d\ level\ of\ concurrency, d = \ maximum\ degree\ of\ concurrency,\ d_o = maximum\ degree\ of concurrency\ for\ the\ smallest\ test\ case.$$

3.2 Capability performance

In addition to the ensemble aspects of MotifNetwork, use of parallel processing is required for the bulk of the analysis. This type of multi-processor usage is called capability computing. In this section are described the mathematical framework, the algorithms, examples of usage and parallel performance results

3.2.1 MotifNetwork: Data products and algorithms

Orchestrating the workflows of MotifNetwork results in several kinds of fundamental data products with some processed results. These results arise from the invocation one of two analysis programs and thus can be classified into one of two groups sets. Each group arises from the processing of the same MotifNetwork intermediate data. These data are contained in the file named ProteinVSMotifs_pseudo_SIF_format_complete (pseudoSIFS) file. This file contains a compact representation of the aggregate set of IPRscan data. A partial example is collected in Table 3. These data are from MotifNetwork results of the genome of Anopheles.gambiae and consists of 122,173 rows. Many times, an Interpro likelyhood score will not be available. All non-available score values are converted to the single value of 'NA'.

Table 3 Example of a (partial) MotifNetwork bipartite (halfpair) data set

ProteinID		Domain ID	eScore	Start Position	End Position

AGAP000002-PA	g2d	noIPR	NA	135	147
AGAP000002-PA	g2d	noIPR	NA	190	209
AGAP000002-PA	g2d	noIPR	NA	217	249
AGAP000002-PA	g2d	noIPR	NA	7	70
AGAP000002-PA	g2d	noIPR	NA	93	242
AGAP000002-PA	g2d	IPR001202	2.8e-11	10	39
AGAP000002-PA	g2d	IPR001202	7.8e-10	9	41
AGAP000002-PA	g2d	IPR001202	NA	14	39

This is a compact representation of a bipartite graph constructed with proteins and domains as vertices. The domains represent *k-mer* entitiies connected to proteinIDs. domainIDs are thus not necessarily unique across proteinsIDs. The term *halfpair* is used to represent one or more elements of this data set. Each of the following processing steps permits the selection of including or not, the noIPR entries form the final analysis. We now describe the details data sets for the ScoreMatrix and WebMatrix applications, respectively.

3.2.1.1 ScoreMatrix

The ScoreMatrix processing of the halfpair data results in two matrices. These matrices are referred to as the Protein_motif_DataBase_matrix and the Protein_motif_Position_matrix. These are two dimensional matrix representations of the biadjacancy graphs that relate proteins (p) to domains (d). The matrix reports the Interpro likelihood scores (eScores) or starting and ending positions on the indicated protein sequence, respectively. We refer to non-nil entries as 'hits'. For a given protein-domain matrix element, it is often the case that multiple hits are identified. This can arise for two reasons. The first is a truly repeated domain. The second reflects hits from multiple Interpro databases. MotifNetwork users have the choice to view all results of just result from one of the set of databases. In any event, the analysis codes restrict the number of identified hits to 10. An example of one of these data sets is collected in Table 4. Several data sets from processed genomes are collected in [51]

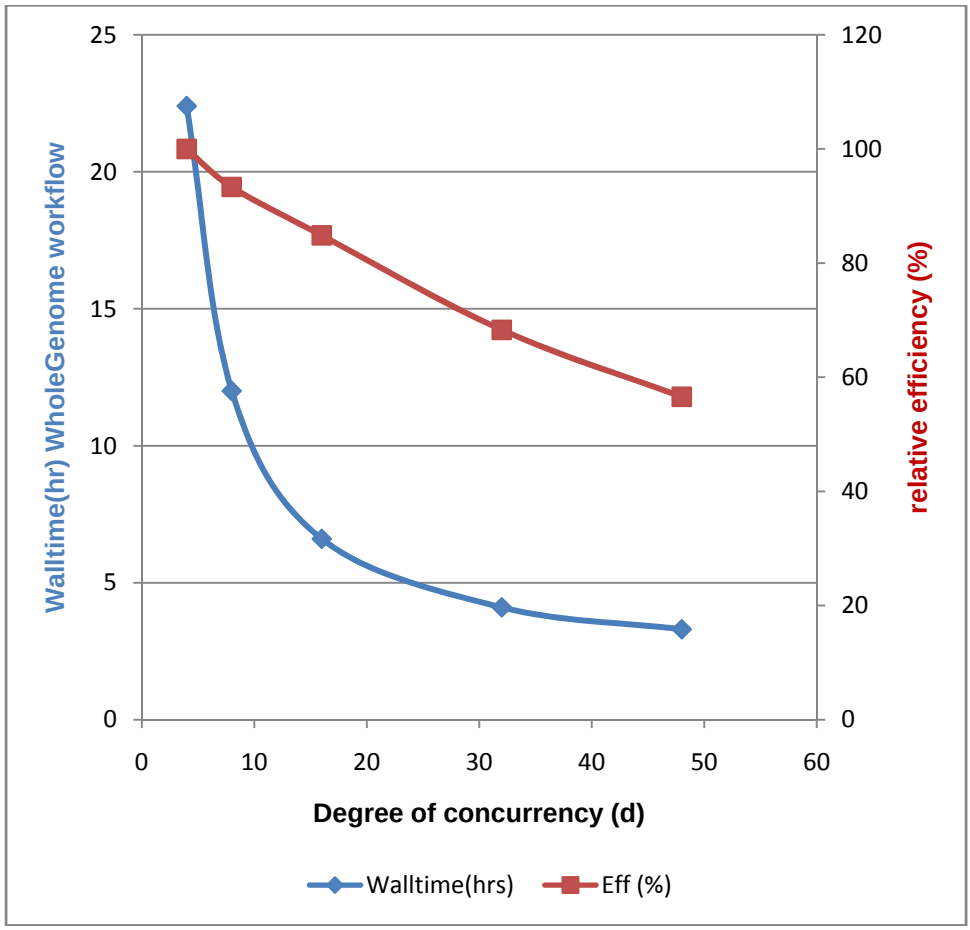

Fig. 8 WholeGenome workflow: performance versus Degree of concurrency. The Walltime (hr) is a measure of the runtime of the workflow. The relative efficiency (%) is the calculated CE(%) value

This is a partial matrix extracted from our Drosophila.simulans results. The actual matrix dimensions were 9,995 proteins connected to 4,714 unique domains. The number of input sequences was 15,415. noIPR results were excluded from the final data set. It is important to note that the MotifNetwork matrices only report proteins that have identified domains. No 'empty' proteins are reported. Lastly, since valid scores can include zero, we use the symbol 'x' to denote a nil entry. Capturing the data into this dense matrix format certainly is not the most efficient use of storage. However, it does facilitate the importing of the text-formatted data into various spread-sheet programs. For many smaller data sets this is a common procedure for MotifNetwork users.

Table 4 Example of the *DataBase* matrix for Drosophila. This is a highly simplified partial representation. The full matrix is 9,995 proteins by 4,714 domains (excluding noIPR results)

Data base/Position	IPR001983	IPR011323	noIPR	IPR000595	IPR003117	IPR0208437	IPR018105
FBpp0208433	5.3000E-25	5.6999E-27	0.0	x	x	x	x
FBpp0223408	x	x	1.5E-25	x	x	x	x
FBpp0208435	x	x	4.4E-7	6.1E-23	5.3E-05	x	x
FBpp0208437	x	x	x	x	x	1.2E-12	x
FBpp0210329	6.0E-69	1.7E-30	0.0	x	x	x	4.9E-16

These matrices are in a form that facilitates several kinds of analysis some of which are now described. In the following we concentrate on the 'database' matrix, though the 'position' matrix could have been used as well.

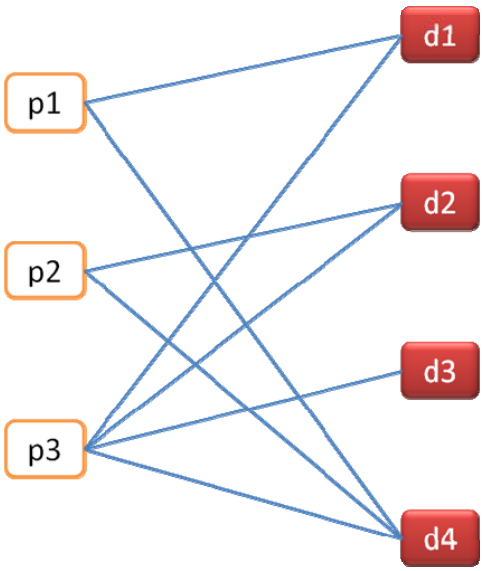

Fig. 9 Illustrative bipartite graph. Unweighted example. See text for more details

The structural form of the database matrix is that of a biadjacency-like matrix except the entries are not integers. Replacing the eScore entries of the Database matrix with an integer indicating the number of hits (or zero) and potentially selecting based on eScore values results in a weighted biadjacency matrix that can be processed. These are sparse matrices with 99.95% nil, 99.94% nil and 99.94% nil for the metazoan genomes H.sapiens. A.gambiae, C.elegens, respectively.

Example

To illustrate the usefulness of these forms, choose a sample system with the following structure. 1s indicate the presence of the domain (d) on the protein (p). 0s indicate the absence of d. This is a simplified (unweighted) system as no duplicate domains are indicated in Table 5 nor graphically in Fig 9.

Table 5 Example (unweighted) MotifNetwork database data structure: matrix form of the halfpair data

protein/domain (**B**)	d1	d2	d3	d4
p1	1	0	0	1
p2	0	1	0	1
p3	1	1	1	1

Let the matrix B represent this unweighted biadjacency matrix. The full adjacency matrix; **A**, becomes

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{0} \end{pmatrix} \quad 2)$$

AA yields a matrix with a very useful form. Specifically, $(A^2)_{ij}$ reports the number of walks of length 2 (*2-walks*) between nodes i and j. These results have a more biological interpretation that is now described. Explicit construction of AA yields Eqn 3. It is a symmetric block diagonal matrix consisting of two submatrices; BB^t and B^tB. These submatrices report complimentary information about the biological system under analysis.

$$\mathbf{AA} = \begin{pmatrix} \mathbf{BB}^t & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^t\mathbf{B} \end{pmatrix} \quad 3)$$

Table 6 Domain pair submatrix (B^tB) example. Derived from the unweighted database data in the previous table

B^tB	d1	d2	d3	d4
d1	2	1	1	2
d2	1	2	1	2
d3	1	1	1	1
d4	2	2	1	3

Using the example data from Table 6, This square domain-domain matrix provides information about domain pairings. For example the diagonal elements (d_i,d_i) report the number of proteins that contain the domain d_i. Also the matrix trace; $Tr(B^tB)$, reports the number of total domains ('hits') found in the system; B. Offdiagonal elements report the number of proteins (p) that **share** the indicated **domain pair** (d_i,d_j). As an example the pair (d_2,d_4) is found on two proteins; p2 and p3. In terms of the equivalent *2-walks* interpretation, (Fig 9), two paths exists between d_2 and d_4. These are d2-p2-d4 and d2-d3-d4, respectively.

Table 7 Protein pair submatrix(BB^t). Derived from the unweighted database data in Table 5

BB^t	p1	p2	p3
p1	2	1	2
p2	1	2	2
p3	2	2	4

The complementary submatrix matrix; BB^t, is collected in Table 7. This reports protein–protein information. The diagonal elements (p_i,p_i) indicate the total number of domains on protein p_i. The offdiagonal elements (p_i,p_j) report the number of **total shared** domains between the two proteins. This may be a value greater than 2 which differentiates the data from Table 6. The matrix trace; $Tr(BB^t)$, again indicates the number of total domains hits in the problem, as expected. As a specific example, the matrix element (p_2,p_3) indicates the two proteins share two domains (d2 and d4). In the associated *2-walks* interpretation, (Fig 9), two paths exist between p_2 and p_3. Namely, p2-d2-p3 and p2-d4-p3, respectively.

Weighted example

Complications in the interpretation of the submatrices arise when the biadjacency matrix; B, is weighted. This is often the case when using multiple Interpro databases on a list of sequences. A slight change in the data of Table 5 serves to illustrate this case.

Table 8 reports a simulated set of weighted values while Fig 10 is the weighted bipartite graph representation. In this data are simulated Interpro reports of 2 instances of d4 on protein p1 and 2 d2 domains on p3.

Table 8 Example (weighted) MotifNetwork database data structure (matrix form of the halfpair data)

protein/domain (**B**)	d1	d2	d3	d4
p1	1	0	0	2
p2	0	1	0	1
p3	1	2	1	1

Evaluate the BB^t submatrix of Eqn 3 Using the values of Table 8 and collecting the results into Table 9. Now consider the effects of the non-unit weights on the final results. The diagonal elements may or may not indicate the total number of domains per protein. From Table 9 element (p2p2)=2 and correctly indicates two distinct connections (domains) on p2 and corresponds to the 2-walks: p2-d2-p2 and p2-d4-p4. Element (p1,p1)=5, however, over-specifies the number of domains. This arises from the 2-walks: p1-d1-p1, p1-d4(2)-p1. But the p1-d4(2)-p1 is of weight 2 and thus yields 4 2-walk combinations. The 4 arises from the 2-walks resulting from a crossover of the lanes.

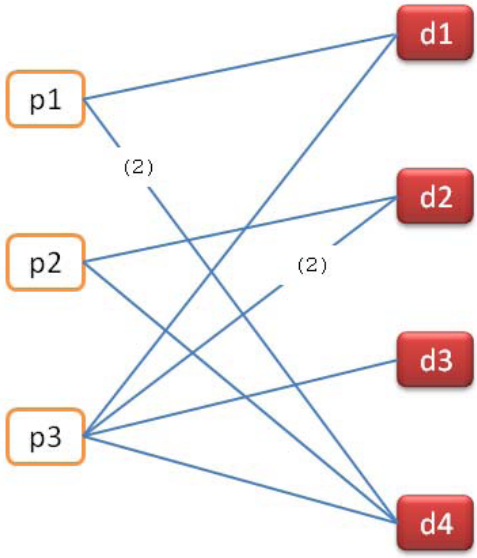

Fig. 10 Illustrative bipartite graph. Weighted example. See text for more details

Table 9 Protein pair submatrix(BB^t) derived solely from the weighted data set

BB^t	p1	p2	p3
p1	5	2	3
p2	2	2	3
p3	3	3	7

To preserve the usefulness of the weighted B matrix in the final results requires a slight change. Let A contain the weighted adjacency data constructed from the elements B. Let A^u be the unweighted adjacency matrix constructed from the unweighted (connection) matrix (C) complement of B.

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{0} \end{pmatrix} \text{ and } \mathbf{A}^u = \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^t & \mathbf{0} \end{pmatrix} \quad 4)$$

To construct a weight conserving 2-walk-like relationship, multiply the two matrices as indicated in Eqn 5.

$$\mathbf{AA^u} = \begin{pmatrix} \mathbf{BC}^t & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^t\mathbf{C} \end{pmatrix} \quad 5)$$

The interpretation of AA^u is a little less obvious than before especially given the resulting matrices are now not symmetric. Let B carry the weighted data as in Table 8. The unweighted data; C, is reported in Table 5. Table 10 reports the results on the mixed submatrix B^tC.

Table 10 Domain pair submatrix (B^tC) derived from the weighted and unweighted forms of the database matrix.

B^tC	d1	d2	d3	d4
d1	2	1	1	2
d2	2	3	2	3
d3	1	1	1	1
d4	3	2	1	4

The matrix trace properly accounts for the total number of domain hits in the system. The diagonal elements themselves also indicate the correct (weighted) number of proteins per domain. From Table 10 element (d_1d_1)=2 that correctly indicates two distinct connections (proteins). These correspond to the 2-walks: d1-p1-d1 and d1-p3-d1. The offdiagonal elements are more difficult to interpret owing to the lack of symmetry. Element (d_4,d_2)=2 while (d_2,d_4)=3. The best way to handle this is to simply take the smallest value; *Vd*, as in Eqn 6 and interpret that as the number of proteins that share the domain pair (ij).

$$Vd = \min\left((\mathbf{BC^t})_{ij}, (\mathbf{BC^t})_{ji}\right) \quad 6)$$

A similar asymmetry occurs for the submatrix BC^t. reported in Table 11.

Table 11 Protein pair submatrix(BC^t) derived from the weighted and unweighted forms of the database matrix.

BB^t	p1	p2	p3
p1	3	2	3
p2	1	2	2
p3	2	3	5

The diagonal elements of this matrix also correctly indicate the (weighted) number of shared domains per protein pair. As in Table 11 some asymmetry can occur. In order to find the correct number of shared domains between (p_ip_j) Compute *Vp* from Eqn 7.

$$Vp = \min\left((\mathbf{B^tC})_{ij}, (\mathbf{B^tC})_{ji}\right) \quad 7)$$

3.2.1.2 WebMatrix data products

The WebMatrix group of results also begin with processing of the halfpair data set. This results in five new data files useful for interpretation of the experiment. They are text formatted files that may be processed by many applications. They are generally formatted, however, to be compatible with the Cytoscape application. Thus, the file extensions are used to denote their cytoscape meaning. Several examples are now addressed for the sample system Anopheles. All images are derived from Cytoscape version 2.4.1.

1. MotifNetwork_WebSummaryData.sif is a sif formatted condensed graph. Nodes indicate domainIDs. Vertices are weighted quantities that indicate the number of proteinIDs the domain pair was found. Fig11 depicts the Summary file for Anopheles. This is screen capture of a Cytoscape graph of 4844 nodes and 33511 edges (partially) displayed in the "Organic" layout. In the CytoPanel is displayed the results of selecting one of the edges using the mouse.This edge connects domains IPR006035 and IPR05924 and indicates that 25 proteins were shared by these 2 domains. To identify which proteins were shared, one can explore the MotifNetwork_WebDetailedData.sif graph.

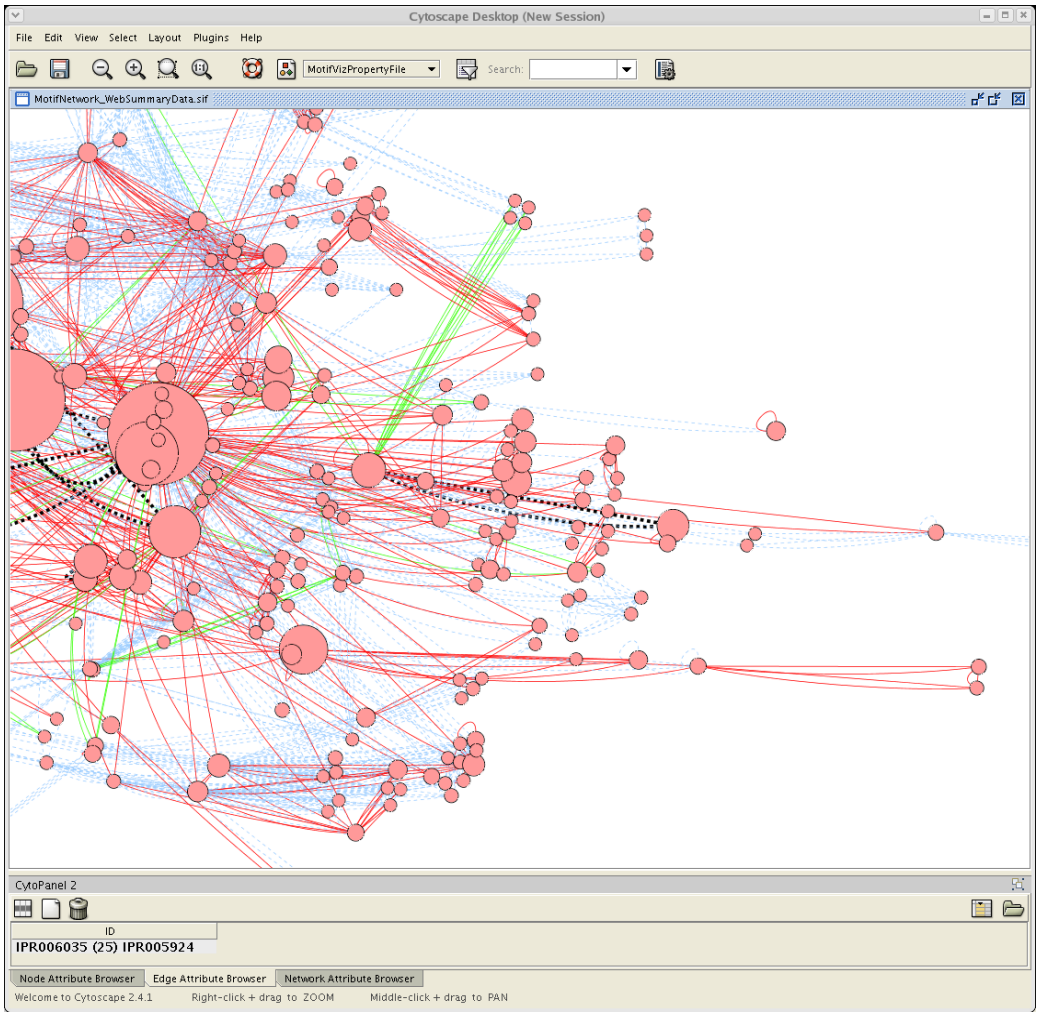

Fig. 11 Screen capture of a Cytoscape session viewing the summary web data. The selection of an edge results in the number of proteins found to contain the corresponding domains (nodes) as indicated

2. MotifNetwork_WebDetailedData.sif. This data set is a sif formatted graph displayed in Fig 12. Graph nodes indicate domainIDs, while edges indicate the specific proteinID on which the connected domains are found. This can be a fairly large graph that carries substantial detail when applied to a full genome. This is screen capture of a Cytoscape (**truncated**) graph of 3,798 nodes and 38,279 edges (partially) displayed in the "Organic" layout. In the CytoPanel is displayed the results of selecting all edges between two nodes of interest. The selected edges are colored red. The results indicate that seven proteins share the two selected domains (IPR006090 and IPR006090) These seven proteins are AGAP00{ 5662,0454,8501,8602,6780,9783,8769}.
3. MotifNetwork_MotifProtein.sif is a sif formatted bipartite graph (Fig 13) the connects domainIDs to proteinIDs. This data set is a sif formatted graph. This is screen capture of a Cytoscape graph of 14,933 nodes and 28567 edges (partially) displayed in the "Organic" layout. In the CytoPanel is displayed the results of selecting one of the edges using the mouse.This edge connects domains IPR006035 and IPR05924 and indicates that 25 proteins were shared by these 2 domains. To identify which proteins were shared, one can explore the MotifNetwork_WebDetailedData.sif graph.
4. MotifNetwork_radii.pvals is a cytoscape attributes file that contains all domainIDs and the number of proteins these IDs were found. The file may be used by cytoscape to generate node radii cues.
5. MotifNetwork_InteractionStrength.pvals. This is an attributes file that specifies vertex weights for the WebSummaryData sif file.

Lastly, a sixth file is made available with all workflow results. This MotifVizPropertyFile.props file is used by Cytoscape to apply the aforementioned attribute files to the sifs graphs. Additional detailed descriptions of some of this data have been reported [16].

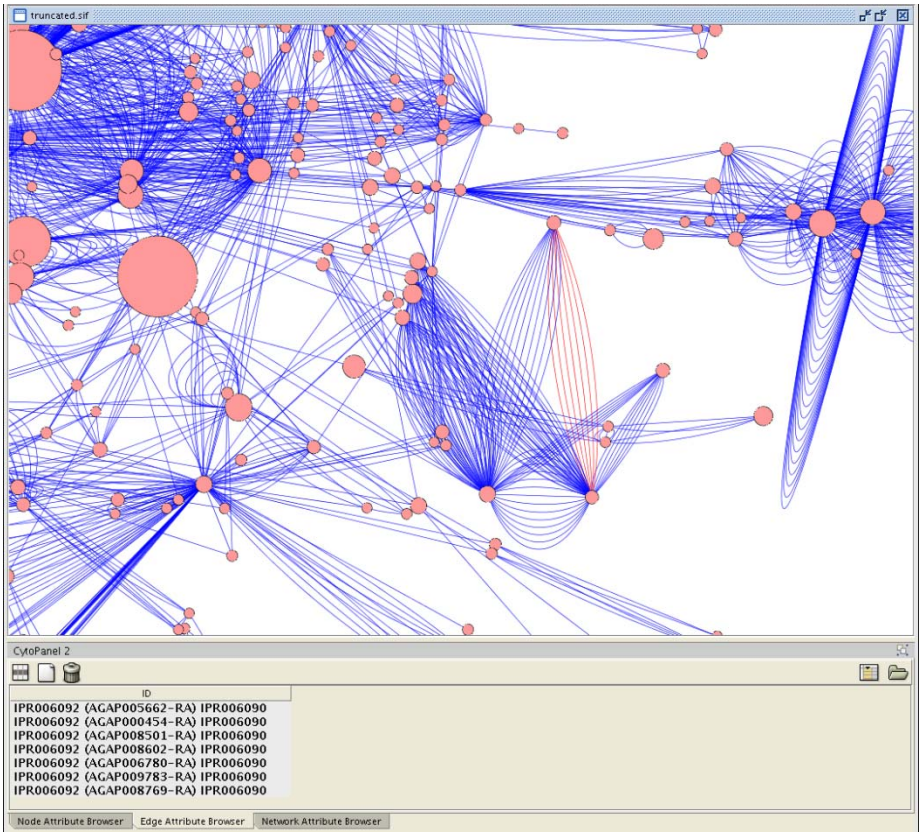

Fig. 12 Screen capture of a Cytoscape session viewing the detailed web data. The selection of an edge results in display of the protein ID that contains the corresponding domains (nodes)

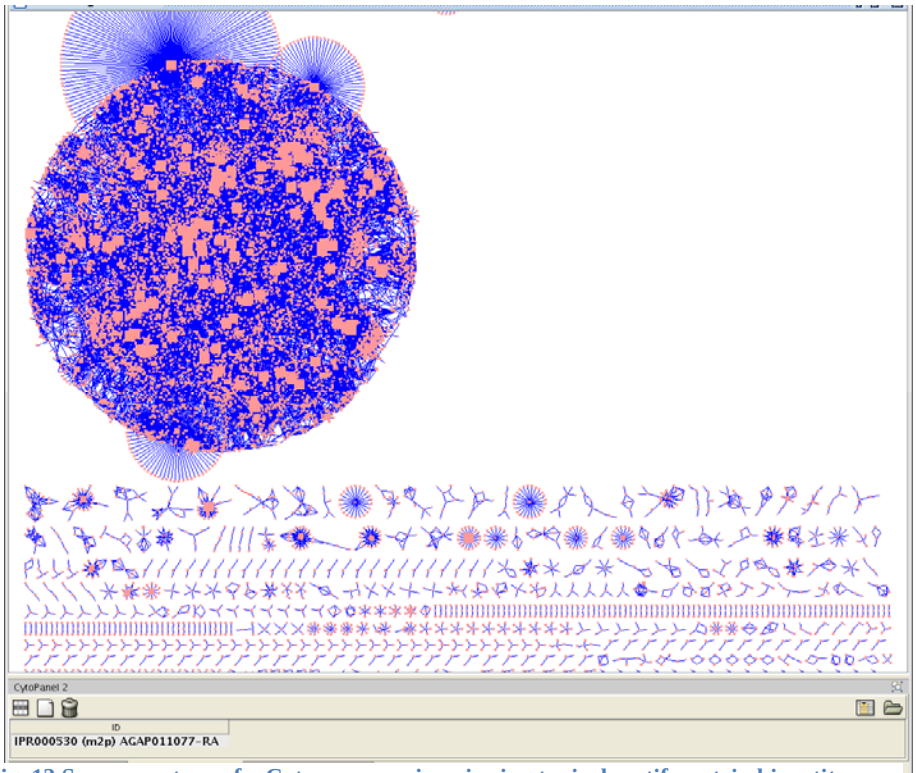

Fig. 13 Screen capture of a Cytoscape session viewing typical motif-protein bipartite data. The selection of an edge results in the identification of a domain found on a protein

3.2 Algorithms and Performance

Generation of the ScoreMatrix and WebMatrix data requires a significant amount of computation. Decreasing the time to solution is accomplished by resorting to parallel processing techniques. To effectively exploit such techniques requires understanding the computational details and characteristics of the computations. These computations are significantly dependant on the placement of domains within the genome.

Thus, understanding the distribution of these domains is vital to developing effective algorithms.

The distribution of unique domains within a genome is highly skewed. The evidence suggests that the distribution follows a power law relationship. This is consistent with the *scale-free* nature of the distribution webs that have been reported [52]. As an example, displayed in Fig 14 is the variation of the number of *Hits* versus the domain index for two genomes. Hits is an indication of the number of proteins where the selected domain were identified. The domain index is an arbitrary index of the unique domains in the genome. These data result from MotifNetwork analysis of the genome: H.sapiens and M. musculus. The following results only include those arising from the PFam [29] dataset.

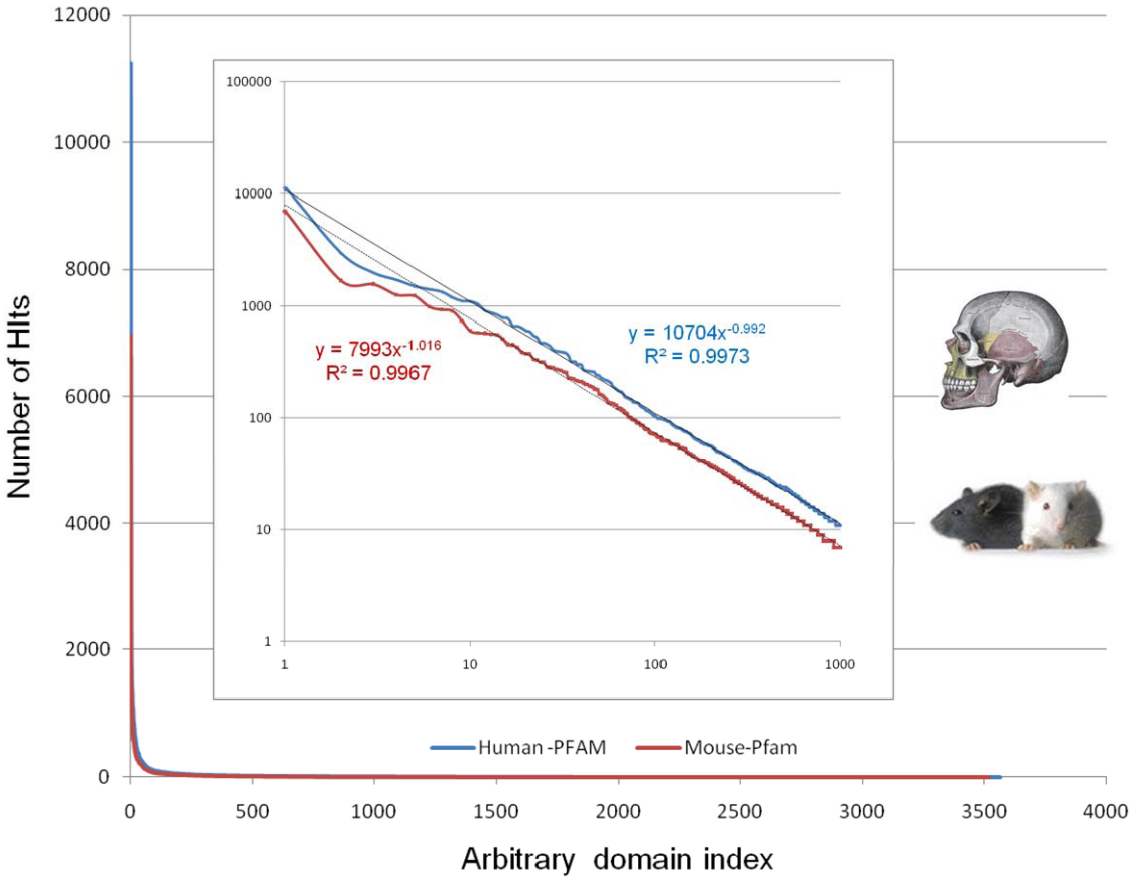

Fig. 14 The determined total number of domain identifications (hits) versus domain index. Index is arbitrary. The data compare results for h.sapiens versus m.musculus.A simple power law fit to the data are indicated. These data use only the PFam components of the Interpro data set

The basic power law-like nature of the data sets is reasonably clear. The data begin to deviate from inverse proportionality at the asymptote. These details, however, impact little the description of the computational load overall. The form of this distribution is used to approximate work distributions in the analysis algorithms.

3.2.1 ScoreMatrix

Construction of the database (position) matrix from the set of halfpairs is now described. Let Sij represent the S matrix element for the ith protein and the jth domain. No domain or protein ID repeating is permitted. V(a) represents the eScore and position values.

$$S_{ij} = \sum_{a}^{T} \left(\delta_{d_j d_a}\right) V(a) \left(\delta_{p_i p_a}\right), \qquad T = Number\ of\ halfpairs \tag{8}$$

For all following Eqns, indexing over α, β indicate scanning through the *halfpairs* list as exemplified in Table 4. Eqn 8 represents several operations. Sij reports the number of times the protein; pi was found with the domain dj. The string matching requirement is represented in Eqn 8 as a dirac delta-like function. In this case returning the value of 1 if dj equals dα, or zero otherwise.

$$S = \sum_{i}^{np} \sum_{j}^{nd} \sum_{a}^{T} \left(\delta_{d_j d_a}\right) V(a) \left(\delta_{p_i p_a}\right) \quad 9)$$

To calculate the entire matrix; S (Eqn 9) requires computing all Sij. np represents the *unique* proteins in the problem while dn the *unique* domains. For a typical genomes np=10-50,000 and nd=4-6,000 unique domains. Values of T can range from 50,000-1,000,000 for whole genomes. In Fig 15 are plotted the value of T for 55 Metazoan genomes available from NCBI. The specific organism names are collected in Appendix 1. These domains were identified using Interpro V19.0 and include all non-commercial databases in the scan.

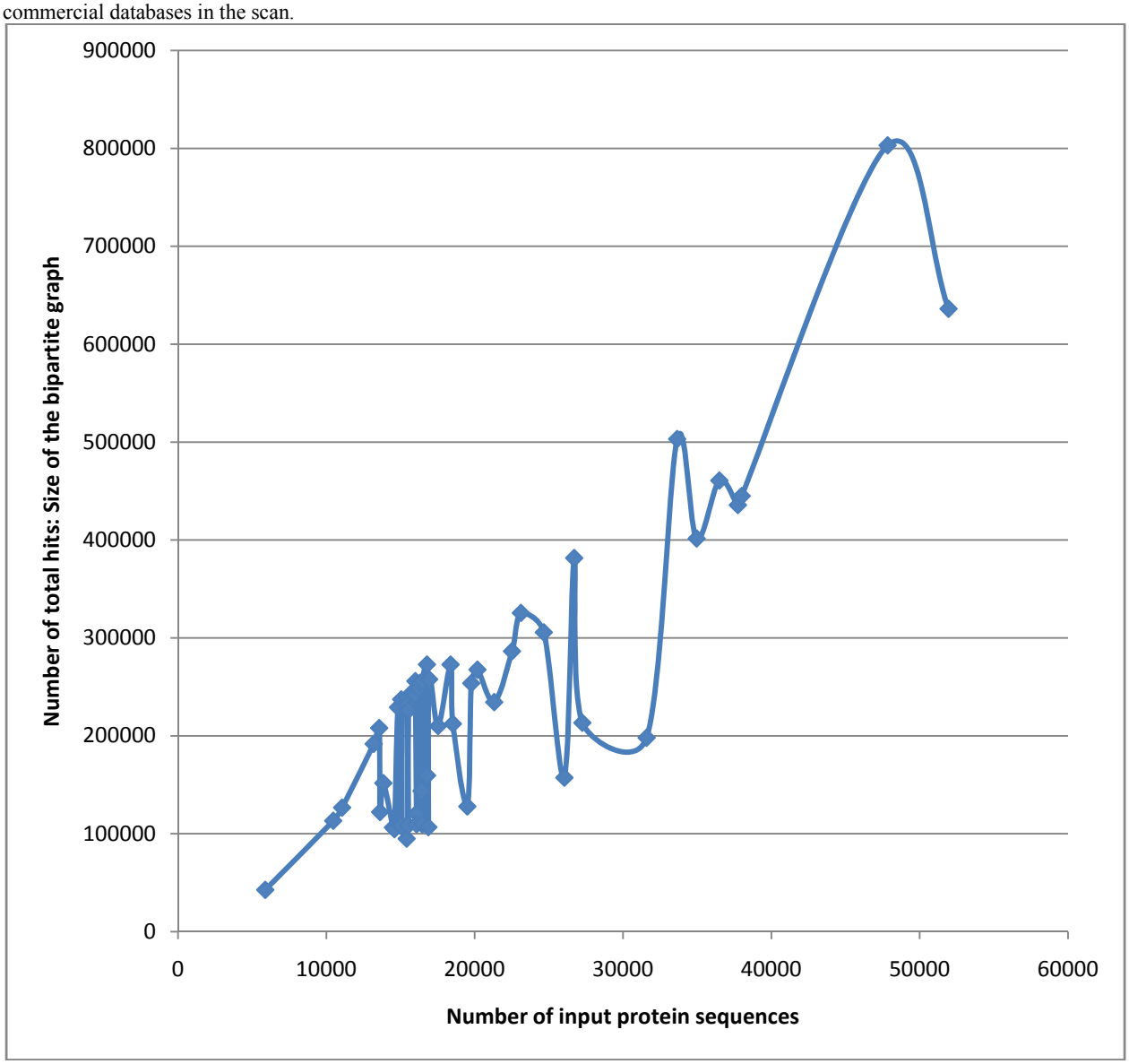

Fig. 15 This is a plot of the total number of bipartite graph elements versus input genome size in input sequences. The data consist of 55 of the 56 total metazoan genomes available [35]. The specific list of genomes and their order are collected in Appendix 1.

One simple decomposition scheme is to treat the vector; Si as a unit of work for a processor. For each i assigned to a processor, compute all j and α-dependent terms. The assignment of Si terms to processors (data decomposition scheme) can be performed statically.

$$S_i = \sum_{j}^{nd} \sum_{a}^{T} \left(\delta_{d_j d_a}\right) V(a) \left(\delta_{p_i p_a}\right) \qquad 10)$$

The benefits to this approach are that the influence of the power-law distribution of domains can be greatly mitigated. The protein-centric work selection results is a fairly balanced distribution of computational load. For a given protein we generally observe at most 1-10 domains per protein regardless of any particular domain's abundance. For a calculation of the H.sapiens data set on a fairly small Dell Linux cluster we plot the total number of domains processed for each processor. Each processor calculated results for many Si. Thus, these are aggregate values per processor. The final data set was of size 25,578 proteins by 6,859 domains (excluding noIPR results). As indicated, the work distribution is balanced as are the total runtimes per processor.

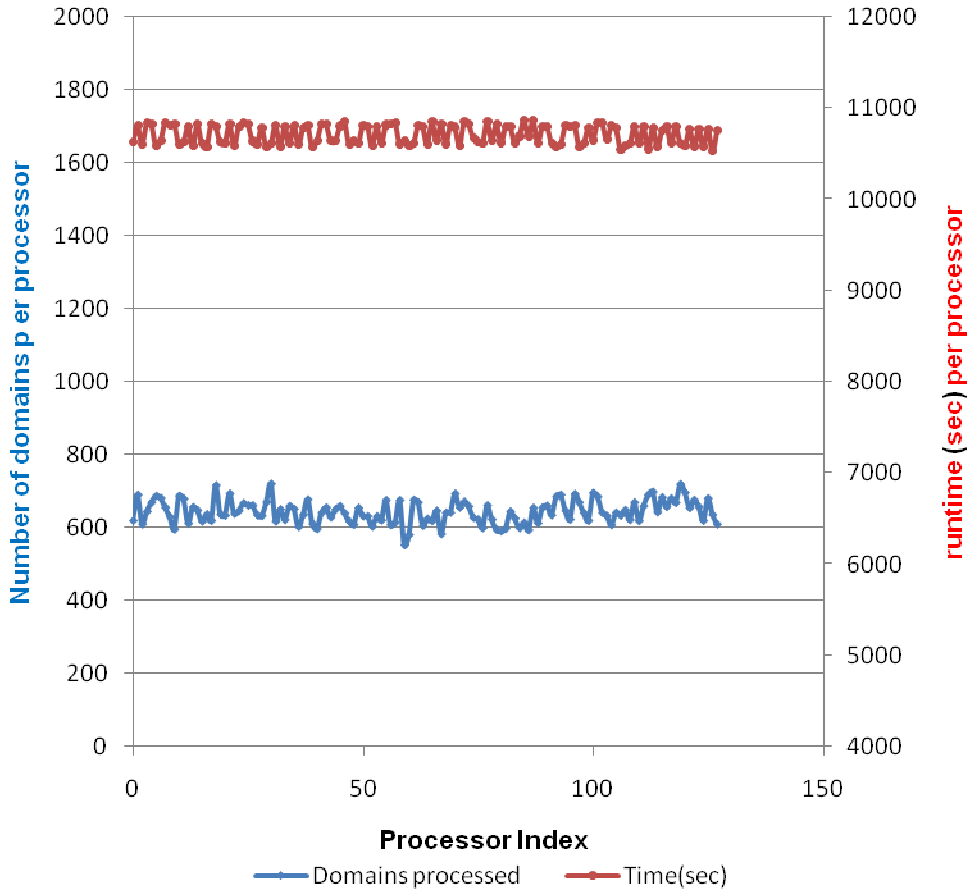

Fig. 16 ScoreMatrix analysis: A measurement of work distribution across processers. The number of domains per processor indicates the resultant number of domains processed by the processor. The runtime is the measured runtime and should be proportional to the work. Data are from Interpro v19.0. Calculations were performed on 128 processors

3.2.1.1 ScoreMatrix algorithms

Construction of the DataBase (position) matrices utilize both the MPI and MPIIO standards. Generally each processor opens and maintains a unique file and constructs the domain contributions for several Si. The final step is a global merging of all current file data into a single matrix with writing of that matrix to a formatted file. This method was chosen because for large genomes (or metagenomes) the data structures simply could not be retained in memory. The format of the final product must be

that of a file so that the workflow/service environment that controls the MotifNetwork orchestration can send the results back to archival storage using efficient network protocol.

Below is listed the ScoreMatrix algorithm pseudocode showing the data decomposition scheme used to implement Eqn 10. This is a simple data parallel approach where each processor (me) performs a subset of the index; i, using a strided access (via i+numCores). The halfPairList data set has been previously processed (condensed) so that multiple entries for a protein-domain pair are combined into a single entry. Red colored type specifies tasks to processors. Me is an integer identifying the processor ID. numCores is the total number of cores.

```
ScoreMatrix decomposition
for (i=me; i<numUniqueProteins; i=i+numCores) {
        proteinWord = UniqueProteinList[i]->protein
        for (j=0;j<numUniqueDomains;j++) {
                domainWordWord = UniqueDomainList[j]->domain);
                for (k=0;k<numHalfPair ;k++) {
                        if (proteinWord = halfPairList[k]->protein) {
                                if ( domainWord = halfPairList[k]->domain ) {
                                        valueScoreWord =halfPairList[k]->eScore);
                                        break;
                                }
                        }
                }
                localScoreMatrixRow[j] = valueScoreWord;
        }
// Construct formatted matrix row
        for (j=0;j<numUniqueDomains;j++) {
                tempPositionRow = tempPositionRow + localScoreMatrixRow[j]
        }
MPI_File_write(ScoreFile, ScoreRow, ScoreLength, MPI_CHAR, MPI_STATUS_IGNORE);
}
```

The resulting set of local (exclusive) MPIIO files now need to be assembled into a single global file that can be archived. This is a conceptually simple process since each processor carries parameters of local file sizes that can be used. The bookkeeping complexity and memory management procedures have been excluded for clarity. The construction of the final data file is coordinated by the `me=0` block of code. Thus it plays a special role with regards to the ordering and writing of the local *score_file* data into the *final_file* file. Parallelism is limited in this routine y the me=0 core routine that simply requests from the other cores (in order) their data contribution. Overall, this step is <1-3% of the total runtime even on large numbers of cores.

```
DataAssembly
if (me == 0) {
        MPI_File_read(score_file, tempSub, localDatasize, MPI_CHAR, MPI_STATUS_IGNORE);
        MPI_File_write(final_file, tempSub, localDatasize , MPI_CHAR, MPI_STATUS_IGNORE);
        for (i=1; i<numCores; i++) {
                MPI_Recv(&NumChars, 1, MPI_INT,i,sizeTag,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                MPI_Recv(tempSub, NumChars,MPI_CHAR,i,lineTag,MPI_COMM_WORLD, MPI_STATUS_IGNORE);
                MPI_File_write(final_file, tempSub, NumChars , MPI_CHAR, MPI_STATUS_IGNORE);
         }
} else {
        MPI_Send(&localsize, 1, MPI_INT, 0, sizeTag, MPI_COMM_WORLD);
        MPI_File_read(score_file, tempSub, localsize, MPI_CHAR, MPI_STATUS_IGNORE);
        MPI_Ssend(tempSub, localsize, MPI_CHAR , 0, lineTag, MPI_COMM_WORLD);
}
MPI_File_close(&score_file);

if (rank==0) {
        MPI_File_close(&final_file);
}
```

3.2.1.2 Performance results

Fig 17 report the total time to solution (tts) for the ScoreMatrix computation as a function of the number of applied cores. These results are for the H.sapiens data set. The first reported results were performed on a Dell Linux cluster comprised of 70 PowerEdge 1955 blades and with an InfiniBand interconnect. Each node contains 2 x 2.66Ghz Intel Woodcrest 5150 (dual core) processors. The program was compiled using the Intel compiler (EM64T icc v 9.1) and using the OpenMPI (1.1.4) implementation of MPI and MPIIO. Also displayed in Fig 17 are tts results from a larger IBM BG/L (BlueGene) parallel machine. The BG/L system was upgraded with 1024 MB memory per (dual core) compute node. Each core is an embedded (ASIC) 700 MHz PPC440 processor. Theoretically, a BG/L 700 MHz core should achieve 2.8 GFlops while an EMT64 core

(2.66 GHz) from our Dell cluster should achieve 10.4 GFlops. This factor of 3.7 manifests itself in the longer runtimes (per core) on the BG/L. The ScoreMatrix code was compiled using the IBM blrts_xlc compiler. Though the machine is slower (per core), good parallel scaling is observed (> 85%, see ahead) indicating a favorable load-balance and small impact of the parallel communications and I/O that is being performed.

As indicated, the program executes well on both computer systems. The time to solution decreases with the number of applied processors. The relative times between the Linux cluster and the BG/L system are consistent with the ratio of processor speeds. For the Linux Cluster plot, we observe that at 256 applied cores the tts is 1.51 hr. Extrapolation (justified by the good observed scaling) back to a single core indicates a time of 386 hrs.

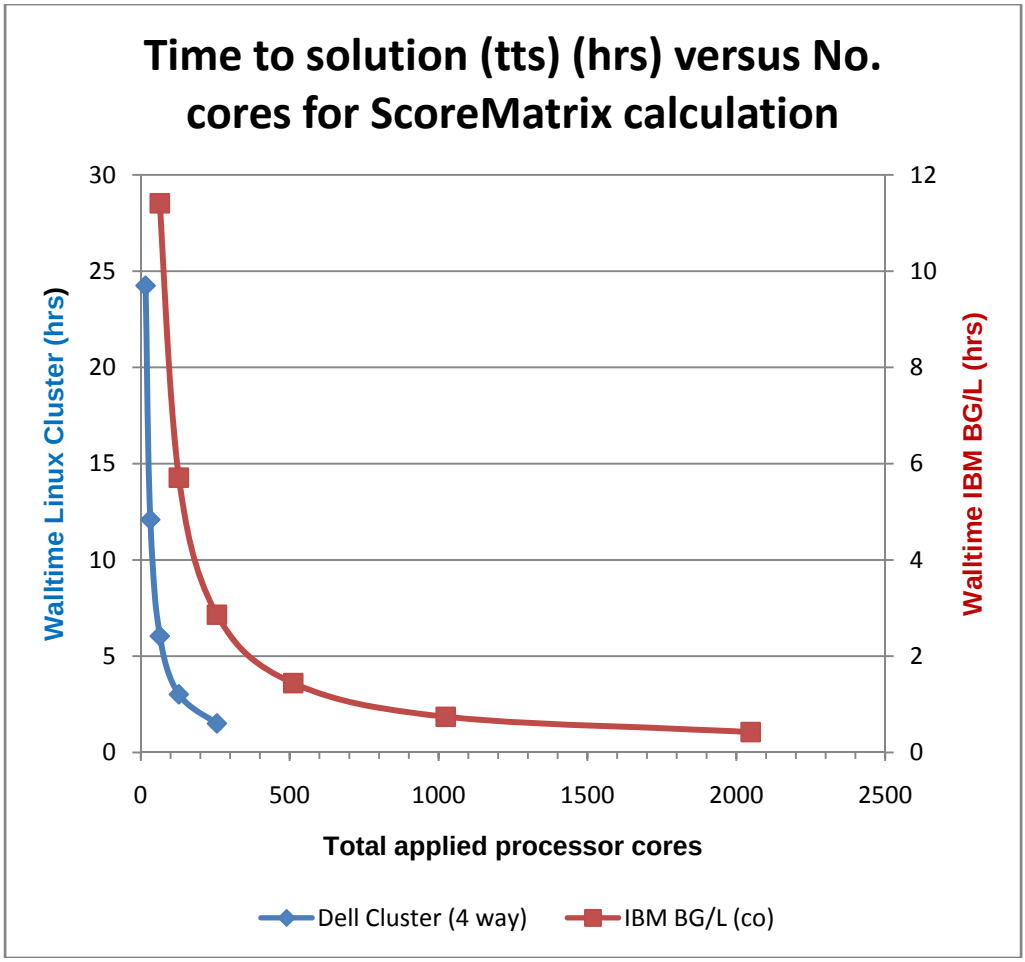

Fig. 17 Total time to solution (tts) (hr) versus the number of applied processor cores for the ScoreMatrix application. Compared are results obtains in a Linux cluster and a larger specialized IBM BG/L system

Fig 18 reports the percent relative speedup (SU) for the ScoreMatrix computation as a function of the number of applied cores for both test computers. This relationship is similar in intent to the relative concurrency (%) used to judge workflow concurrency efficiency. These are measures of parallel efficiency for the program. The percent relative speedup is a quantity that measures the ability of a parallel program to effectively utilize more cores for a given problem size (*hard scaling*). Owing to the length of these computations, no run could be performed on a single core. Thus, the reported SU(%) values are based on the results at the smallest set of cores (p_o).

Execution of the ScoreMatrix application on both machines indicates a better than 85% parallel efficiency over the range of applied cores. Ideal scaling would report 100% SU(%). For the smaller Linux cluster, SU(%) remain at near 100% for all tests. Only at very large numbers of cores (> 1024)(BG/L) does the value drop below 90%.

$$SU(\%) = \frac{100 \times t(p_o) \times p_o}{t(p) \times p} \qquad 11)$$

$t(p) = walltime\ on\ p\ cores, p = \ number\ of\ applied\ cores,\ p_o\ = minimum\ num.\ cores$

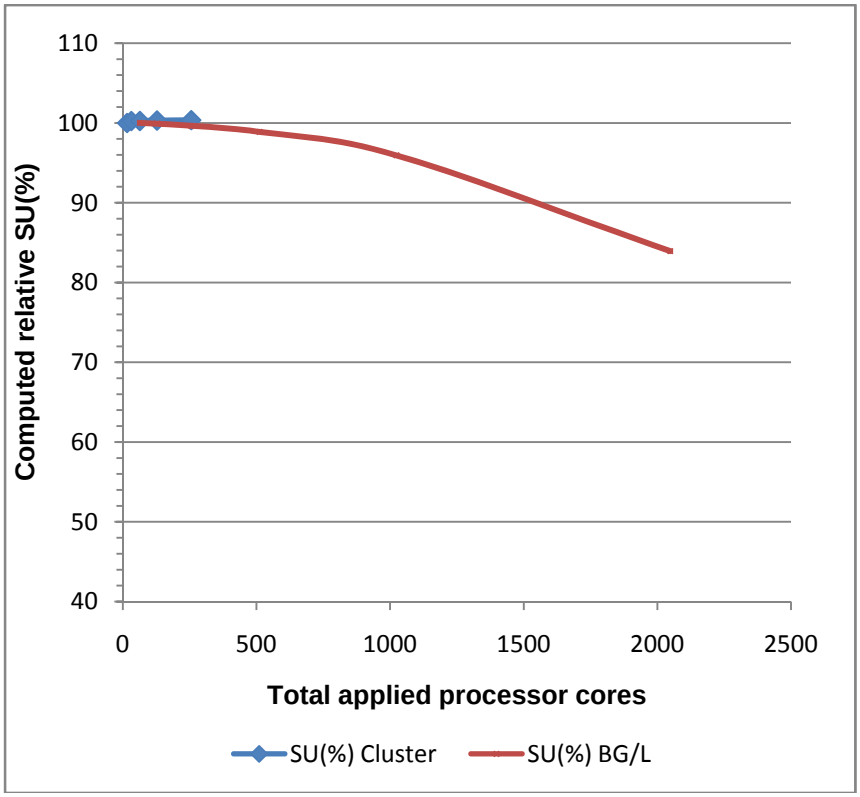

Fig. 18 Computed relative speedup, SU(%), for versus the number of applied processor cores for the ScoreMatrix application. Compared are results obtains in a Linux cluster and a larger specialized IBM BG/L system

3.2.2 WebMatrix Data mathematical detail

Computation of the domain-domain detailed and summary graphs is very sensitive to the domain distribution characteristics indicated in Fig 14. and thus creating a fast algorithm is slightly more complicated. The reason for this is, in contrast to the ScoreMatrix computations, it is not possible to cast these problems into processing lists of proteins without resorting to large amounts of interprocessor communications.

The basic equation driving the generation of the domain-domain detailed webs is listed as Eqn 12. In short, this function sweeps through the halfpair list and constructs the triplets di-pa-dj. The detailed web calculation seeks to process the halfpair data set to generate all domain-domain pairs that share a protein. This is a doubly nested loop over unique domains and sweeps through the halfpair file in a nested fashion.

$$Dweb\ = \sum_{i}^{nd}\sum_{\alpha}^{T}\left(\delta_{d_i d_\alpha}\right)\sum_{j}^{nd}\sum_{\beta}^{T}\left(\delta_{d_j d_\beta}\right)\left(\delta_{p_\alpha p_\beta}\right) \qquad 12)$$

In short the first double summation first processes all unique domains in the problem (nd), and finds instances of each domain on the half-list group; T. This can be simplified somewhat. We are interested in graphs when the shared domains are different. Thus we can condense Eqn 12 to Eqn 13 and exclude d_i,d_i pairs.

$$D\ = \sum_{i}^{nd}\sum_{\alpha}^{T}\left(\delta_{d_i d_\alpha}\right)\sum_{\beta}^{T}\left(1-\delta_{d_\alpha d_\beta}\right)\left(\delta_{p_\alpha p_\beta}\right) \qquad 13)$$

Parallel processing/detailed graphs

To process this graph it is desirable to leverage parallel processing for a faster time to solution (tts). The question becomes how to decompose the problem over large numbers of processors and in a manner that minimizes the amount of interprocessor communication. The first approach chosen was a simple decomposition based on domain index; i. We call this the *static* method. Here the work is divided based on i. The measure of processor work becomes W_i,.

$$W_i = \sum_{\alpha}^{T} \left(\delta_{d_i d_\alpha}\right) \sum_{\beta}^{T} \left(1 - \delta_{d_\alpha d_\beta}\right) \left(\delta_{p_\alpha p_\beta}\right) \quad 14)$$

This would be a desirable approach given that no data dependencies exist between the i terms. However, owing to the characteristics of the domain distribution, the load-balance across processors is expected to be poor. Consider the first term of Eqn 9 and the results of Fig 14.

$$S_i = \sum_{\alpha}^{T} \left(\delta_{d_i d_\alpha}\right) \equiv P(i) \quad 15)$$

The summation over α sweeps over all halfpairs looking for matches to the domain di. According to Fig 14, The number of di, varies approximately as a simple power-law function; P. Thus Eqn 15 shows a work equivalence relationship between the halfpair scanning the domain distribution function; P. The first summation for Wi, varies as P and the double summation then varies approximately as P^2. It is unlikely that any kind of task aggregation scheme based on i can balance this load over large numbers of processors.

A more complicated distribution comprised of a finer grain decomposition based on αand using dynamic selection schemes within the algorithm itself can lead to much better balance across processor. This adds the complication, however, of related i data being distributed around multiple processors. This issue doesn't influence directly the detailed web construction but it significantly infuences the summary web construction requiring more interprocessor communication. Define the new work task as Wiαas in Eqn 16.

$$W_{i\alpha} = \sum_{\beta}^{T} \left(1 - \delta_{d_\alpha d_\beta}\right) \left(\delta_{p_\alpha p_\beta}\right) \quad 16)$$

These smaller tasks; $W_{i\alpha}$, continue to vary in effort consistent with the domain distribution;P. However, they are much smaller in total effort (versus W_i) and the number of terms is on the order of n*P. Using a suitable dynamic allocation of $W_{i\alpha}$ to processors, results in overall better load balance characteristics.

3.2.2.1 WebMatrix algorithms

Two algorithms were created based on the task decomposition schemes of Eqn 14 and Eqn 16. One using a simple domain-based decomposition (STATIC) and the other exploiting memory caching and with more complicated communications patterns and dynamic load-balancing (ALT100) using the NXTVAL concept [53] as implemented using MPI. Below are collected the WebMatrix algorithms (pseudocode). Each approach generates local files that must be assembled into the final dataset. The previously described DataAssembly code is reused for this purpose.

Displayed is the detailed web construction pseudocode describing the STATIC data decomposition scheme (Eqn 14). This algorithm requires that the halfPairList data be previously sorted by domains. In this algorithm, a simple data decomposition on the i index is used. The construction of the webDetailed array is used by the subsequent summary web construction program. As a pseudocode representation a significant amount of memory control, string and character processing and timers are omitted. Also omitted are code for handing self-loops that are generated when a domain is only found by itself. Red colored type specifies tasks to processors.

```
for (i=me;i<numUniqueDomains;i=i+numCores) {
        domainFirst = UniqueDomains[i];
        for (j=0;j<numHalfPairs;j++) {
                domainSecond = halfPairList[j]->domain;
                if (domainFirst = domainSecond) {
                        proteinFirts = halfPairList[j]->protein;
                        for(k=0;k<numHalfPairs;k++) {
                                proteinSecond = halfPairList[k]->protein;
                                if( proteinFirst = proteinSecond) {
                                        domainSecond = halfPairList[k]->domain;
                                        if ( domainFirst != domainSecond ) {
```

```
                                        webDetailed[incWeb]->entry,domainFirst,
                                                            proteinFirst, domainSecond);
                                        bufWord = domainFirst+proteinFirst+domainSecond
                                        MPI_File_write(file, bufWord, wordLength, MPI_CHAR,
                                                            MPI_STATUS_IGNORE);
                                        incWeb++;
                                    }
                                }
                            }
                        }
                    }
                }
```

The next block of pseudocode depicts the dynamic task allocation scheme used to implement the results of Eqn 16. This is the ALT100 algorithm. This algorithm requires that the halfPairList data be previously sorted by domains. In this algorithm, we resort to the NXTVAL-like functional call that guarantees the return of a unique value (or set of values if chunk >1) across processors. An MPI implementation of NXTVAL has been described in details at [54]. Much of the client-server code used to manage the NXTVAL counting has been omitted except as indicated. The construction of the webDetailed array is used by the subsequent summary web construction program. As a pseudocode representation a significant amount of memory control, string and character processing and timers are omitted. Also omitted are code for handing the self-loops that are generated when a domain is only found by itself. The DataAssembly program is used to process the data files. Red colored type allocates tasks to the processors.

```
MPE_Counter_create( MPI_COMM_WORLD, &smaller_comm, &counter_comm );
for (i=0;i<numUniqueDomains;i++) {
        domainFirst = UniqueDomains[i];
        for (j=0;j<numHalfPairs;j++) {

                indexjob = (i * numHalfPairs) + j;
                if (value == indexjob ) {

                        domainSecond = halfPairList[j]->domain;
                        if (domainFirst = domainSecond) {
                                proteinFirts = halfPairList[j]->protein;
                                for(k=0;k<numHalfPairs;k++) {
                                        proteinSecond = halfPairList[k]->protein;
                                        if( proteinFirst = proteinSecond) {
                                                domainSecond = halfPairList[k]->domain;
                                                if ( domainFirst != domainSecond ) {
                                                webDetailed[incWeb]->entry,domainFirst,
                                                proteinFirst, domainSecond);
                                                bufWord = domainFirst+proteinFirst+domainSecond
                                                MPI_File_write(file, bufWord, wordLength, MPI_CHAR,
                                                                        MPI_STATUS_IGNORE);
                                                incWeb++;
                                                }
                                         }
                                }
                        }
                        MPE_Counter_nxttask(counter_comm, &value, numsize-1,rank);
                }
        }
}
```

The parameter that drives the chunking of tasks is buried in the MPE_Counter_nxttask method. The C code for this routine is specified below. The value of chunking is provided by the global parameter; `NXTVALCHUNK`.

```
void MPE_Counter_nxttask(MPI_Comm counter_comm, int * value , int ncores , int rank )
{
    int ichunk=NXTVALCHUNK;
    int temp;
    static int icount=0,nleft=0;
    int tempcount=0;
    int newvalue;
    int zero=0;

        if(ncores >= 0) {
                if (nleft == 0)  { //get a new starting value for the iterations
                        MPE_Counter_nxtval(counter_comm, &temp);
                        icount = temp * ichunk;
                        nleft = ichunk;
                }
                *value = icount;
                icount = icount + 1;
                nleft = nleft - 1;
        } else {
                nleft = 0;
                *value = 0;
                MPE_Counter_nxtval(counter_comm, &newvalue);
        }
    return;
}
```

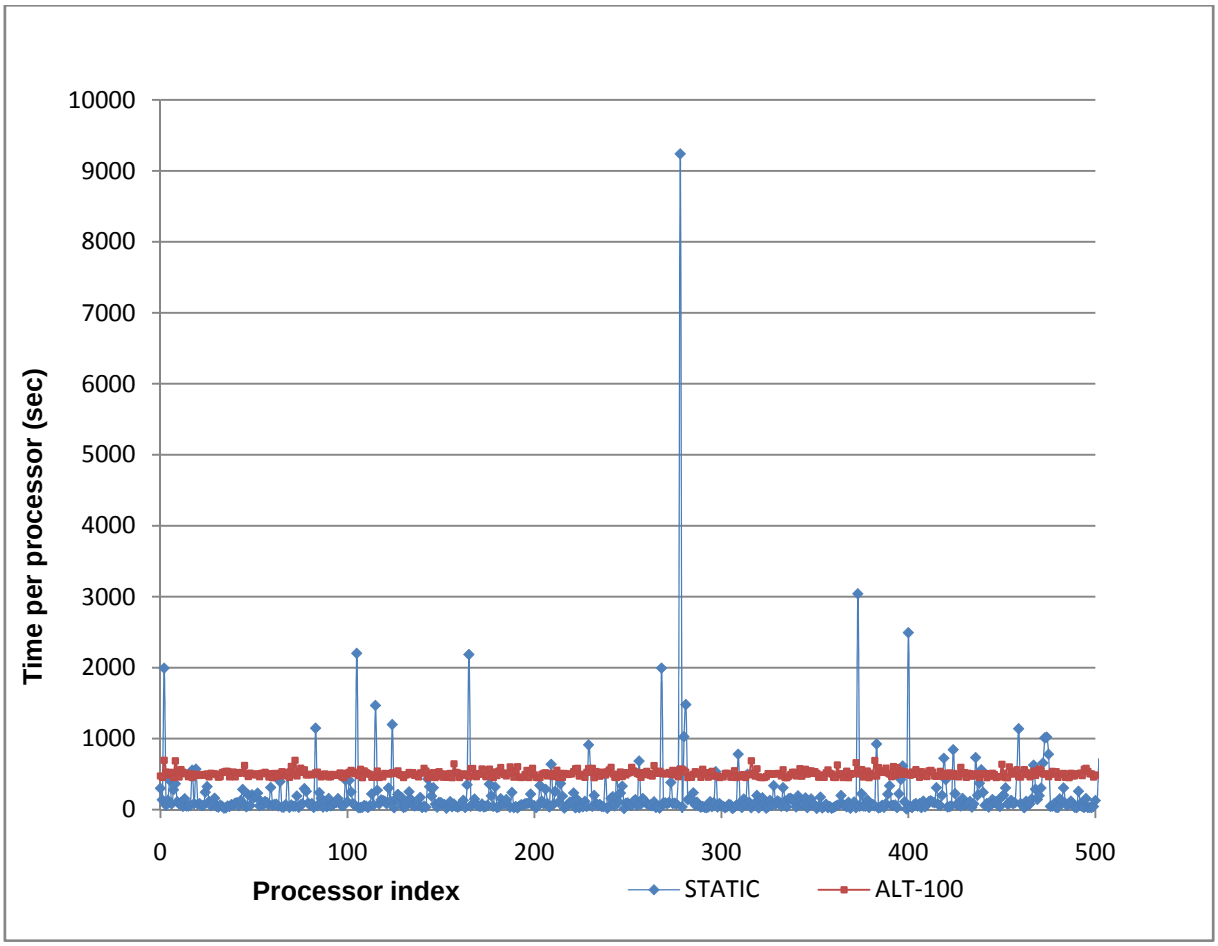

Fig. 19 Measured time versus processor on the IBM BG/L system. Compared are the STATIC and ALT100 algorithms

A demonstration of the load balancing characteristics between the STATIC and ALT100 algorithms are compared in Fig 19. Here are represented work distributions for a calculation of the *detailed* data set for the Anopheles data set. The value of T was 122,173 elements. The calculation was performed using an IBM BG/L computer with 512 applied processor cores. Reported are the total computational times per core for the computations. The STATIC algorithm (static, blue) is based task decomposition using using Eqn 14. The ALT100 algorithm (alt100, red) uses a dynamic selection of tasks based on Eqn 16.

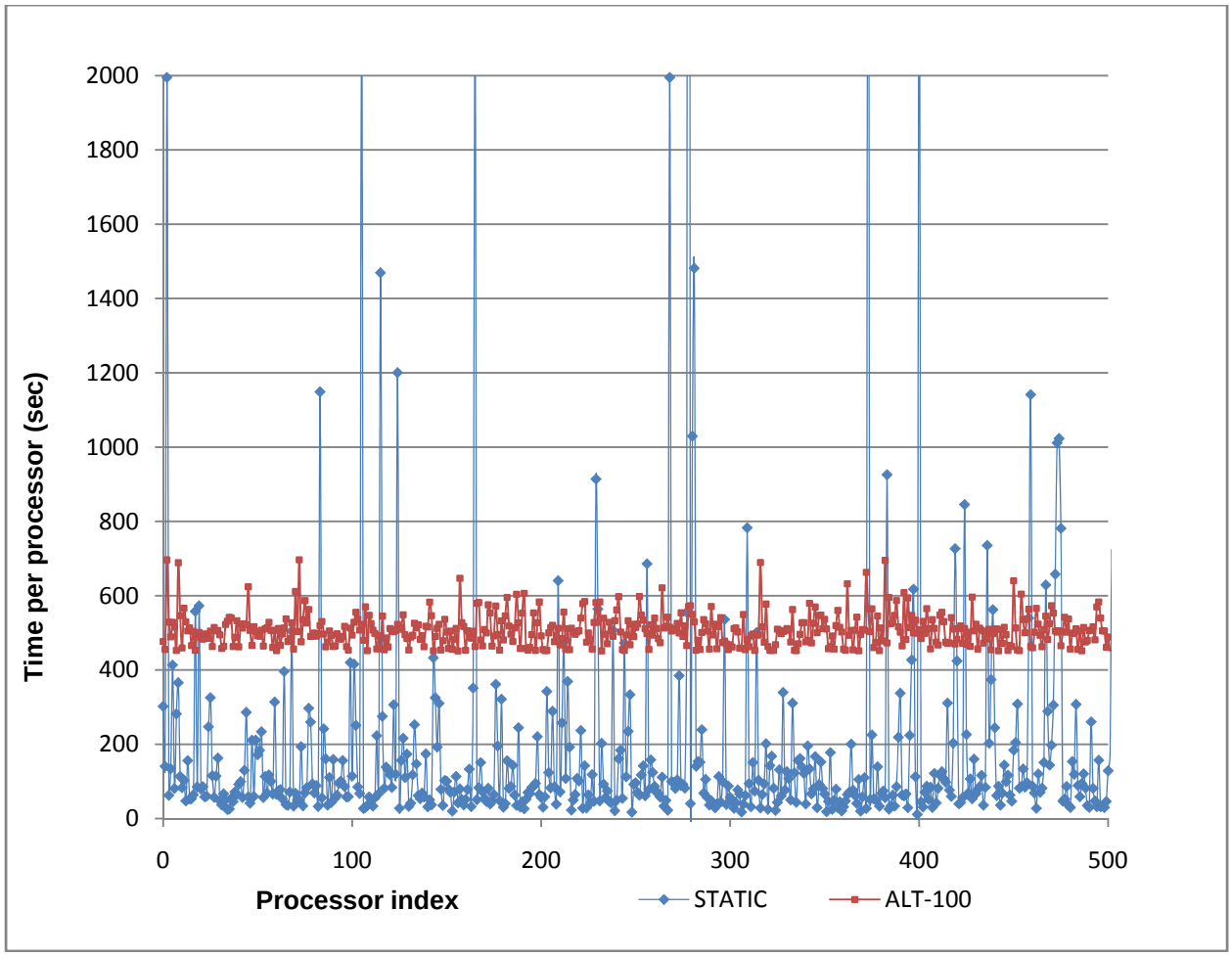

Fig. 20 Measured time versus processor on the IBM BG/L system . Compared are the STATIC and ALT100 algorithms. The details and better work distribution of the ALT100 algorithm is apparent

The *static* results are highly variable across processors ranging from a few tens of seconds to nearly 10,000 secs. This load imbalance severely degrades parallelism resulting in the total time to solution being no less than the greatest single processor time (9,447 secs). The alt100 approach is much more amenable to balance. The overall time per processor increases relative to the static calculation, however, mostly due to interprocessor communication and an increase in the algorithm complexity. To quantify this, the summation of all processor times (from Fig 20) for the static method yields 94,468.9 processor-secs. The alt100 algorithm equivalent becomes 259,016 processor-secs, nearly 3 times greater. Nevertheless, the alt100 algorithm is much faster overall owing to the favorable load balancing. In particular 865 secs versus 9,447 secs, nearly a factor of 10. In the next section, we compare performance and relative SU(%) for these two approaches.

A benefit of the static algorithm is the overall speed **if** conditions are found where load balancing is possible. We compared the number of tasks per processor (proportional to time) of the static approach for several differently sized parallel runs. The results are collected in Fig 21. In this plot are overlaid results for calculations performed on 16, 32, 64, 128, and 256-way partitions. The number of tasks per processor varies widely across processors and for all runs even for the 16-way calculation. Clearly the static task distribution scheme simply cannot overcome the power-law distribution of the domains. For each color-coded run, the task distribution contains hugely dominated by a single (or two) processor. In each case, the total time to solution is dictated by this dominant processor. Thus this domain-based decomposition is just not useful under any foreseeable circumstances.

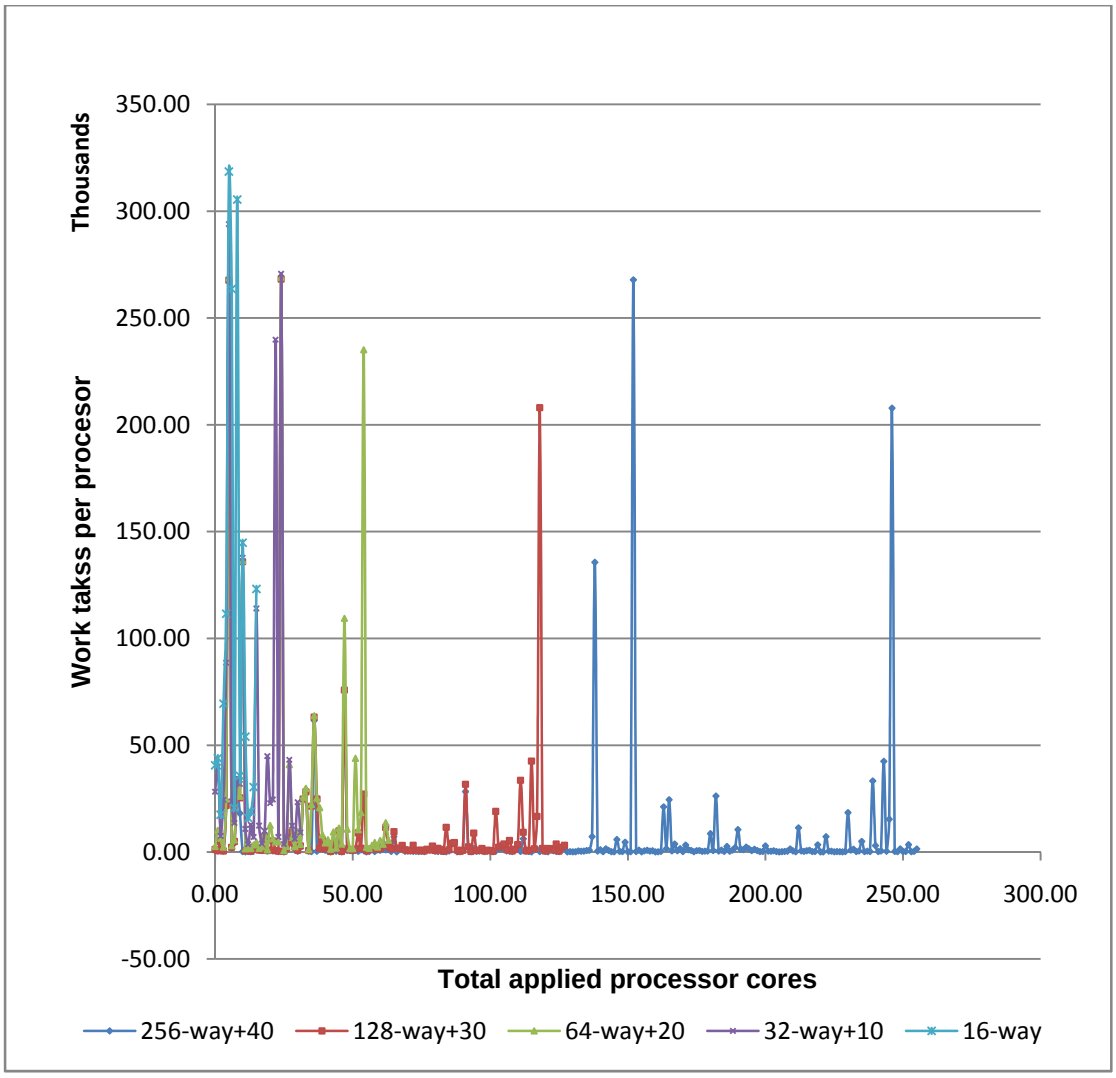

Fig. 21 Measured time versus processor on the IBM BG/L system . Compared are the STATIC and ALT100 algorithms. The details and better work distribution of the ALT100 algorithm is apparent

3.2.2.2 WebMatrix performance and parallel efficiency (SU(%))

Three quantities are generated by the WebMatrix analysis: detailed graphs, summary graphs, and the motifprotein bipartite graph. To demonstrate the impact of the static and alt100 task decomposition schemes on a real problem, the time to construct the detailed graphs and total runtime are plotted as a function of the number of applied cores in Fig 22. Generally, we find that the static results are poorly scaling (poor load balanced as expected) but worse so on the larger core-counts possible on the BG/L. The alt100 algorithm scales much better overall.

In Fig 23, are reported the tts for the total walltime and the detailed graph construction time as a function of the number of applied processors. Both detailed curves are highly scaling decreasing from 912 sec to 83 sec (Linux) and 2,017 sec to 546 sec on the BG/L. The walltime tracks closely the time for the detailed graphs. Noteworthy is the deviation from idea for the Linux results. This degradation from ideal is caused by the summary computation which is scaling poorly on the Linux cluster. Based on the BG/L results, the summary computation can scale quite well for large numbers of processors. Thus it is surmised that the poor scaling on the Linux cluster was caused indirectly by heavy disk I/O by other users on the system. A comparison of the summary, detailed, and motifprotein tts is assembled in Fig 24. The summary computation is quite modest and scales well on the BG/L system. Clearly, the summary computation on the Linux cluster is scaling poorly and in contrast to all the others. The overall scaling is sufficient to reasonably apply hundreds of processors to the problem resulting is a solution hundreds of times faster than on a single core.

To illustrate the parallel efficiency of the WebMatrix analysis, the relative percentage speedup for the walltime and detailed web times are plotted in Fig 25. The efficiency overall is not as good as for the ScoreMatrix results, however, the runtimes are significantly faster than the ScoreMatrix times and the algorithm,m is much more complicated and fluctuations in networking and disk I/O availability. All SU(%) terms are computed from Eqn 11.

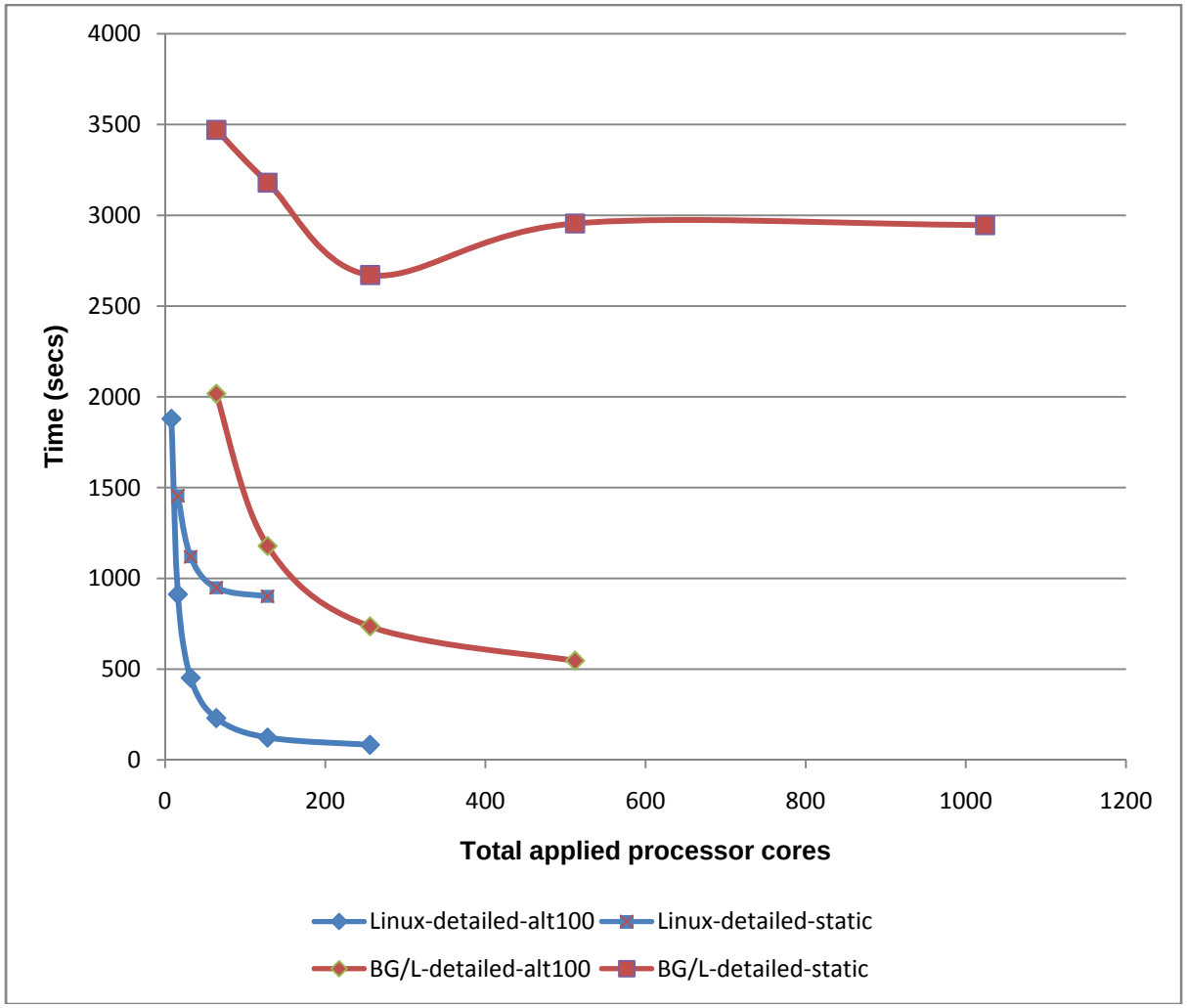

Fig. 22 Time to solution (sec) for performing the detailed web construction. The STATIC and ALT100 algorithms are compared. Calculations were performed on a Dell Linux cluster and the IBM BG/L

The SU(%) terms for the detailed method begins at near 100% but rapidly falls to near 70% (Linux) or at greater processor counts to near 45% (BG/L). These parallel efficiency values reflect the substantial amount of MPI and MPIIO related activities in the algorithm. The walltime SU(%) for the BG/L tracks closely to the detailed values indicating good overall scaling. The walltime SU(%) for the Linux values is much below the detailed values. This is caused by the summary portion of the analysis scaling so poorly for these examples.

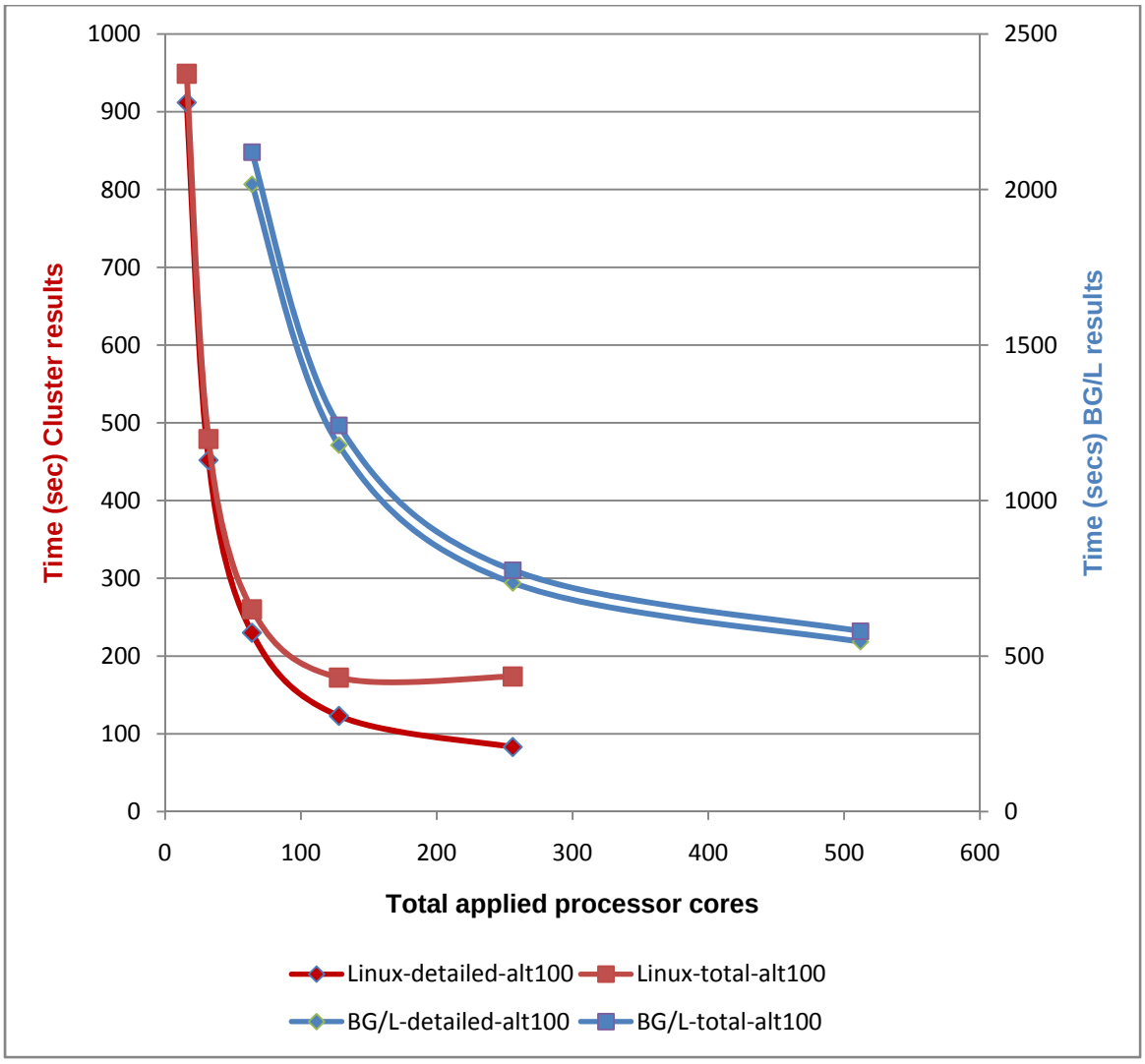

Fig. 23 Comparison of the total walltime (sec) versus and the time to construct the detailed web (sec) versus the number of applied processor cores. The ALT100 algorithm was used. Calculations were performed on a Dell Linux cluster and the IBM BG/L

3.2.2.3 Chunk optimization

Lastly, the construction of the *detailed* graphs is dependent on a chunking parameter. This chunking parameter adjusts the task selection process. Using the NXTVAL concept, each processor would acquire a single task (single α; Eqn 14) to process. Adding the chunking terms agglomerates tasks into larger working chunks. For the reported performance measurements that parameter had been set to 100 tasks at a time (alt100). In Fig 26 are data supporting this selection. A reasonable selection of the value for that parameter is important as it influences the ability to dynamically load balance the work tasks from Eqn 11 versus the additional algorithm overhead relative to the static algorithm. A small value results in more and smaller tasks that can be dynamically load-balanced (better backfilling). The smaller values, however, result in a greater number of function calls and more MPI communications. The calculations used a core-count of 128. The tts solution at value=1 is larger than the runtime of the *static* algorithm indicating the much greater overhead for this algorithm. However, as the value is increased the total tts decreases significantly reaching an asymptote of approximately 123 secs. The smallest value that achieves the greatest benefit occurs at approximately 100. Thus this value has been chosen for all the reported calculations.

3.3 Performance Conclusions

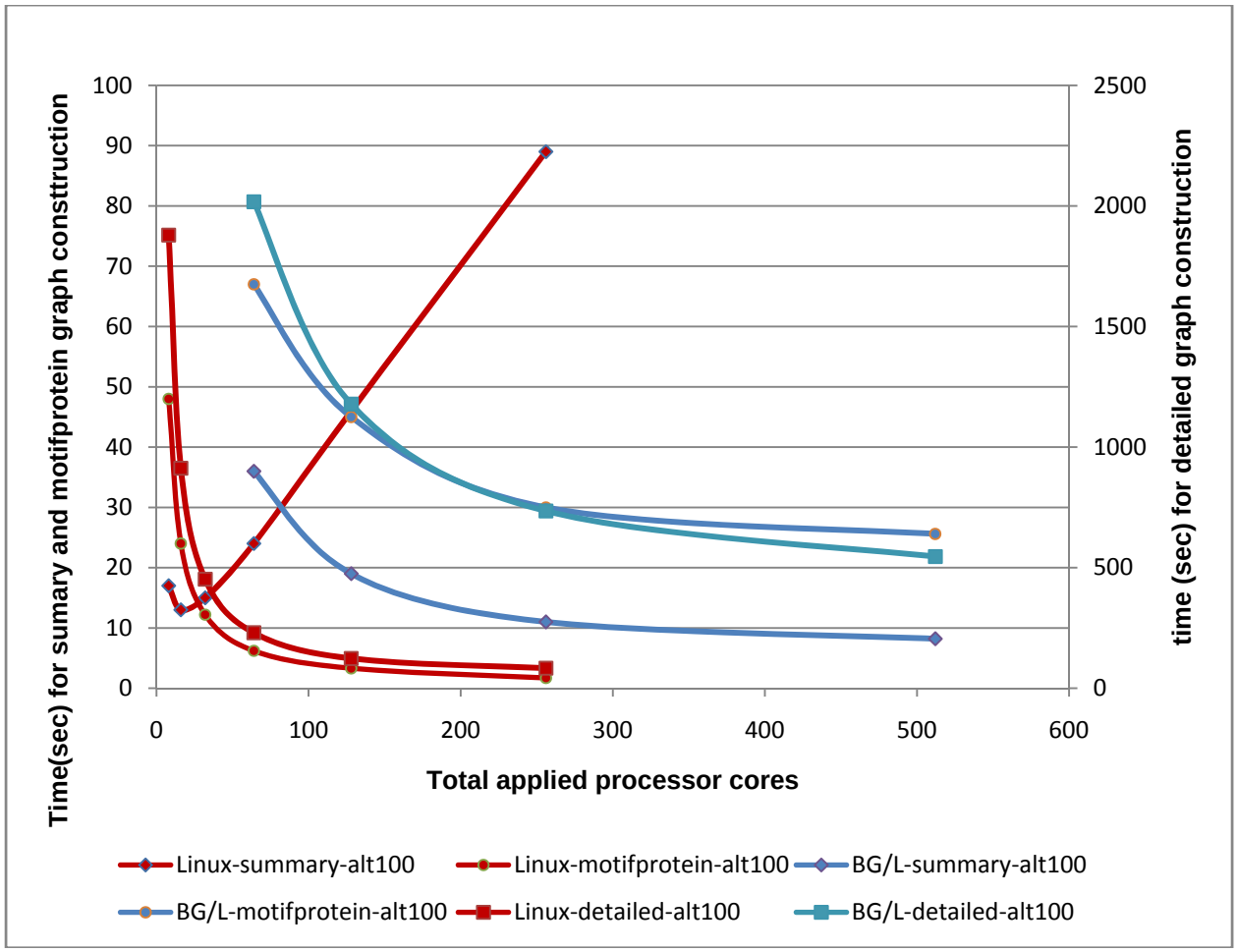

Fig. 24 Comparison of the time to solution of the summary, detailed and motifprotein graphs versus the number of applied processor cores. The ALT100 algorithm was used. The detailed times are 10X greater than the summary and motifprotein times. Calculations were performed on a Dell Linux cluster and the IBM BG/L.

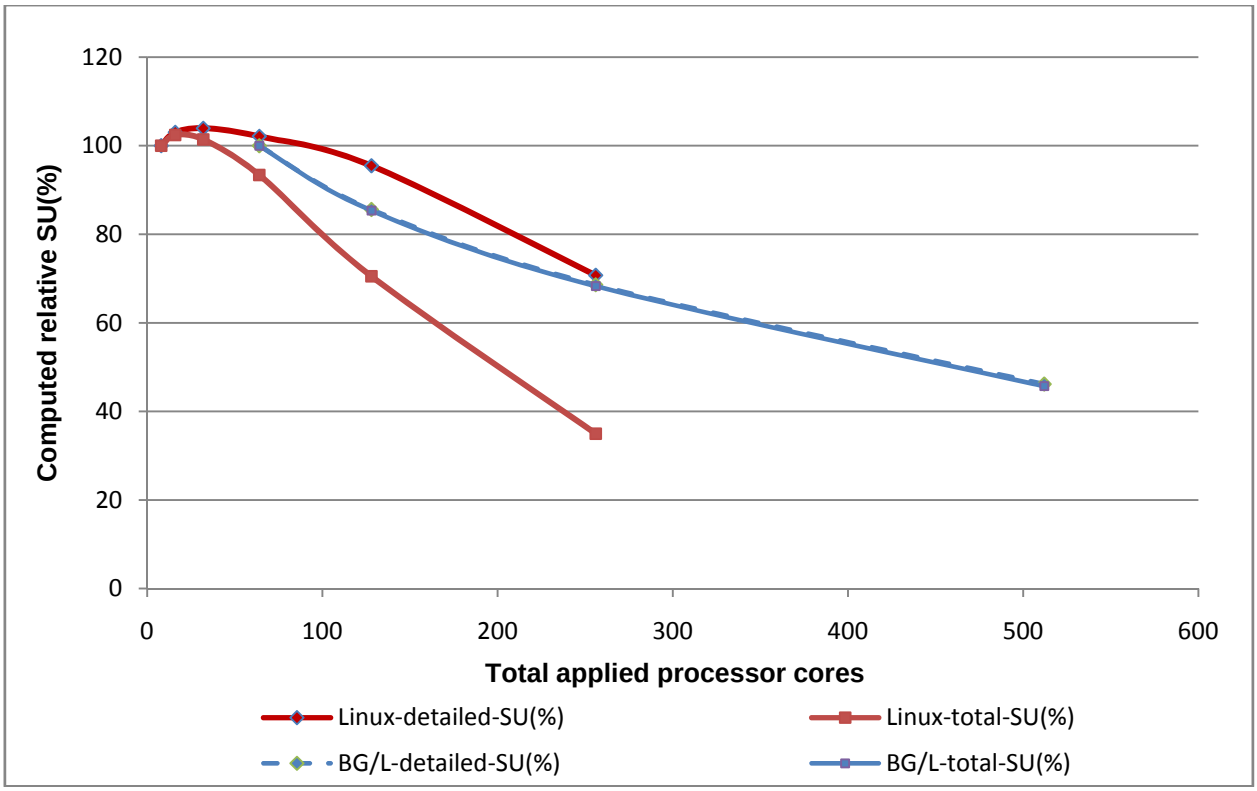

Fig. 25 Relative speedup, SU(%) versus processor cores for the walltime (sec) and detailed web construction time (sec). The ALT100 algorithm was used. Calculations were performed on a Dell Linux cluster and the IBM BG/L.

The processes and parallel computing to speed up analysis. Both aspects have been presented and show good scaling. Ensemble (workflow) runtimes decrease from approximately 90 hr (single core estimate) to 3.4 hr at 48-way concurrent which is a CE(%) of approximately 60%. This is a good result given the measured times include inherently non-scalable (and optional) steps such as the occasionally lengthy (10-15 mins) iRODS replication operation.performance of MotifNetwork is substantially driven by both ensemble computing where domains are scanned by concurrent

A suitable mathematical framework and algorithm have been adopted for performing the ScoreMatrix and WebMatrix calculations. For a typical large genome (H.sapiens) consisting of 37,374 input sequences from the metazoan set of 55 processed genomes that get split into 1,259 chunks, resulting in 25,578 proteins connected to 6,859 domains (excluding noIPR results). The use of parallel processing constructs results in significant decreases in overall time to solution. Table 12 reports observed times versus estimated single core times for the H.sapiens total analysis.

Table 12 Comparison of runtimes of the ScoreMatrix and WebMatrix services for the h.sapiens genome.

Cores (Linux)	ScoreMatrix(sec)	WebMatrix(sec)	Total(hrs)
256	5,436	173	1.5
1	1,396,592	15,032	392

4 MotifNetwork conclusions

The usefulness of MotifNetwork in supporting scientific research is based, in part, on the speed of obtaining results. In this section are reported performance measurements for computationally significant parts of the analysis computations. Some early MotifNetwork performance measurements have also been reported [50]. Some of the analysis procedures depend significantly on the distribution of domains within the data set. The existence of a power-law distribution of domains results in challenges to performing these computations in parallel.

These fast runtimes are a result of the integration of many levels of grid- and distributed-computing, web services, and parallel processing to achieve a throughput of 1,000s of sequences per hour per TFlops of computing. The system is scalable to both larger genomes and multiple researchers competing for access. The ultimate goal of processing a genome per day has been accomplished. The system is now being incorporating into a larger comparative analysis framework.

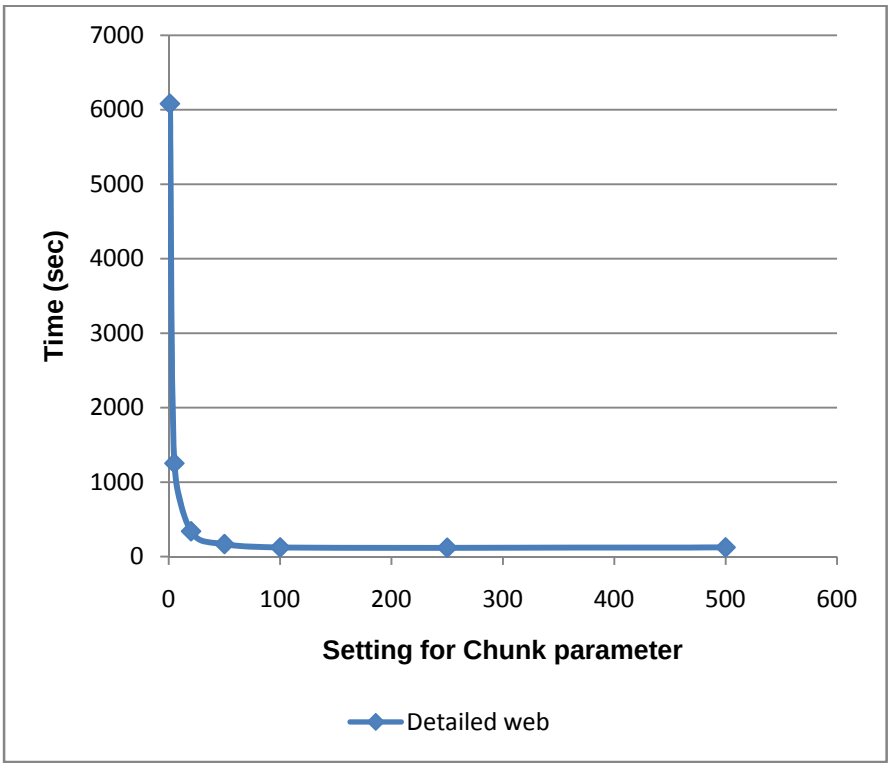

Fig. 26 Preliminary optimization of the ALT100 chunking parameter. Plotted are the time to perform the detailed web construction versus chunking parameter. All calculations were performed on 128 cores on the Dell Linux cluster. The input data are from H.sapiens

Acknowledgements

Support from the National Science Foundation under Grant No. (0835651) and the Renaissance Computing Institute are greatly appreciated. In addition, a particular thanks to the following professionals that assisted this making this system a reality. Mr. Bradley Viviano (Globus), Dr. Gopi Kandaswamy (gstLite), and Ms. Leesa Brieger (iRODS).

References

[1] J.L. Tilson, G. Rendon, E. Jakobsson, "MotifNetwork: High throughput determination of Evolutionary Domain Network" Proceedings of the 2009 International Conference on Bioinformatics and Computational Biology (BIOCOMP'09), July 13-16, 2009, USA (to appear).
[2] Gene Ontology, "The Gene Ontology Consortium Gene Ontology: tool for the unification of biology". *Nature Genet.* 25: 25-29 (2000).
[3] V. Kunin, I. Cases, A.J. Enright, V. de Lorenzo, and C.A. Ouzounis, "Myriads of protein families, and still counting". *Genome Biol* 200, 4:401 (2003).
[4] A. Tasneem, L.M. Iyer, E. Jakobsson, and L. Aravind, "Identification of the prokaryotic ligand-gated ion channels and their implications for the mechanisms and origins of animal Cys-loop ion channels," *Genome Biol*., 6(1), R4 (2005).
[5] N. Bocquet, L. Prado, J. Cartaud,J. Neyton, C. Le Poupon, A. Taly, T. Grutter, J.P. Changeux , PJ. Corringer, "A prokaryotic proton-gated ion channel from the nicotinic acetylcholine receptor family", *Nature* 445, 116-119, 2007.
[6] Ger M-F. In preparation
[7] R.F. Doolittle "The multiplicity of domains in proteins", *Annu Rev Biochem*, 64, 287-314 (1995).
[8] C.A. Orengo and J.M. Thornton, "Protein families and their evolution – a structural perspective", *Annu Rev Biochem*, 74, 867-900 (2005).
[9] G. Apic, J. Gough, S.A. Teichmann, "Domain combinations in archaeal, eubacteria, and eukaryotic proteomes" *J. Mol. Biol.* 310, 311-325 (2001).
[10] J.H. Fonga, L.Y. Geera, A.R. Panchenkoa, and S.H. Bryant. "Modeling the Evolution of Protein Domain Architectures Using Maximum Parsimony" *Journal of Molecular Biology*,366(1), pp 307-315 (2007).
[11] A.K. Bjorklund, D. Ekman, et al. "Domain Rearrangements in Protein Evolution." *Journal of Molecular Biology* 353(4): 911-923 (2005).
[12] P. Kersey and R. Apweiller, "Linking publication, gene and protein data", *Nature Cell Biology*, 8(11) (2006).
[13] I. Foster, C. Kesselman, Chapter 2 of *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman, 1999.
[14] Teragrid science gateway developments as supported by the National Science Foundation Office of Cyberinfrastructure, grant number 1234567 "ETF Grid Infrastructure Group: Providing System Management and Integration for the TeraGrid" and by grant number 7654321 "SCI: TeraGrid Resource Partners.", www.teragrid.org.
[15] The OpenScienceGrid (OSG) (www.opensciencegrid.org).

[16] J.L. Tilson, G. Rendon, M.-F. Ger, E. Jakobsson, "MotifNetwork: A Grid-enabled Workflow for High-throughput Domain Analysis of Biological Sequences: Implications for annotation and study of phylogeny, protein interactions, and intraspecies variation" in *7th IEEE International Conference on Bioinformatics and Bioengineering* (BIBE'07), 2007, 620-627.

[17] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience*, vol. 18, iss. 10, pp. 1067-1100, 2006.

[18] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, D.S. Katz, "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems" *Scientific Programming Journal*, Vol 13(3), pp. 219-237, 2005.

[19] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock. *System demonstration, 16th Intl. Conf. on Scientific and Statistical Database Management* (SSDBM'04), 21-23 June 2004, Santorini Island, Greece. http://www.kepler-project.org/

[20] Majithia S. Shields M.S.,Taylor I.J. and Wang I. *Triana: A Graphical Web Service Composition and Execution Toolkit.* In Proceedings of the IEEE International Conference on Web Services (ICWS'04), pages 514-524. IEEE Computer Society, 2004. http://www.trianacode.org/

[21] The Workflow Emulator. http://www.dataandsearch.org/provenance/

[22] J.L. Tilson, M.S.C. Reed, and R.J. Fowler. "Workflows for performance evaluation and tuning". In *Proceedings 2008 IEEE International Conference on Cluster Computing (Cluster 2008)*, page 8pp, Tsukuba, Japan, September 2008. IEEE.

[23] E.M. Zdobnov and R. Apweiler, "InterProScan – an integration platform for the signature-recognition methods in Interpro" Bioinformatics, 17(9), 2001, 847-848.

[24] R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher, L. Cerutti, F. Corpet, M. D. Croning, R. Durbin, L. Falquet, W. Fleischmann, J. Gouzy, H. Hermjakob, N. Hulo, I. Jonassen, D. Kahn, A. Kanapin, Y. Karavidopoulou, R. Lopez, B. Marx, N. J. Mulder, T. M. Oinn, M. Pagni, F. Servant, C. J. Sigrist, E. M. Zdobnov, "The InterPro database, an integrated documentation resource for protein families, domains and functional sites," *Nucleic Acids Res*. 29:37-40, 2001.

[25] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks", *Genome Res*., Nov; 13: 2498 – 2504, 2003.

[26] S. F. Altshul, T. L. Madden, A. A. Schaffer, J. H. Zhang, Z. Zhang, W. Miller, D. J. Lipman "Gapped BLAST and PSIBLAST: a new generation of protein database search programs," *Nucleic Acid Res*, 25: 3389-3402, 1997.

[27] T.K. Attwood, P. Bradley, et al. "PRINTS and its automatic supplement, prePRINTS." Nucl. Acids Res. 31(1): 400-402, 2003.

[28] C. Bru, E. Courcelle, et al. "The ProDom database of protein domain families: more emphasis on 3D." *Nucl. Acids Res*. 33(suppl_1): D212-215, 2005.

[29] R.D. Finn, J. Mistry, et al. "Pfam: clans, web tools and services." *Nucl. Acids Res*. 34(suppl_1): D247-251, 2006.

[30] I. Letunic, R. R. Copley, et al. "SMART 5: domains in the context of genomes and networks." *Nucl. Acids Res*. 34(suppl_1): D257-260, 2006.

[31] N.J. Mulder, R. Apweiler, et al."New developments in the InterPro database." Nucl. Acids Res. 35(suppl_1): D224-228, 2007.

[32] D. Wilson, M. Madera, et al. "The SUPERFAMILY database in 2007: families and functions." *Nucl. Acids Res*. 35(suppl_1): D308-313, 2007.

[33] N. Hulo, A. Bairoch, et al. "The PROSITE database." *Nucl. Acids Res. 34*(suppl_1): D227-230, 2006.

[34] The Taverna workbench manual v1.7.1. http://taverna.sourceforge.net/.

[35] Several genomes were referred to in this report. These include: Homo Sapien (EBI), Taeniopygia guttata (Sanger), Anopheles gambiae (P3.4, Anopheles gambiae Genome Project). In addition the complete set of 56 metazoan genomes (including E.coli W3110) were downloaded from NCBI and processed.

[36] A. Datta, J.L. Tilson, G. Rendon, E. Jakobsson, " A High-throughput sialylmotif analysis in the glycosyltransferase protein family", Proceedings of the annual TeraGrid '09 conference (TG09), Arlington (VA), June 22 - 25, 2009.

[37] L. Ramakrishnan, D. Gannon, "A Survey of Distributed Workflow Characteristics and Resource Requirements", Technical Report TR671, Sept 2008, Department of Computer Science, Indiana University,Bloomington.

[38] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon "Building Web Services for Scientific Grid Applications," *IBM Journal of Research and Development*, 50(2/3), pp. 249-260, 2006.

[39] G. Kandaswamy, and D. Gannon, "A Mechanism for Creating Scientific Application Services on Demand from Workflows," 2006 International Conference on Parallel Processing Workshops (ICPPW'06), pp. 25-32, 2006.

[40] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" International J. Supercomputer Applications, 15(3), 2001.

[41] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd ed., MIT Press, 1994.

[42] G.von Laszewski, I. Foster, and J. Gawor, "CoG kits: a bridge between commodity distributed computing and high-performance grids" in Proceedings of the ACM 2000 Conference on Java Grande (San Francisco, California, United States, June 03 - 04). JAVA '00, 2000. ACM Press, New York, NY, pp. 97-106.

[43] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2006, pp 2-13.

[44] W. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal S. Tuecke, S., "Data Management and Transfer in High Performance Computational Grid Environments," Parallel Computing, 28 (5), pp. 749-771, 2002. gram

[45] I. Foster, C. Kesselman,"Globus:Metacomputing Infrastructure Toolkit," International Journal of Supercomputer Applications, 11(2):115-128, 1997.

[46] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," Fifth ACM Conference on Computer and Communications Security, pp. 83-92. 1998.

[47] J. Novotny, S. Tuecke, V. Welch, "An Online Credential Repository for the Grid MyProxy", Tenth International Symposium on High Performance Distributed Computing (HPDC-10), 2001.

[48] European Bioinformatics Institute www.ebi.ac.uk/InterProScan

[49] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, "A Prototype Rule-based Distributed Data Management System", High Performance Distributed Computing (HPDC) workshop on "Next Generation Distributed Data Management", May 2006, Paris, France.

[50] J. L. Tilson, A. Blatecky, G. Rendon, M.-F. Ger, E. Jakobsson, "MotifNetwork: Genome-Wide Domain Analysis using Grid-enabled Workflows" in *7th IEEE International Conference on Bioinformatics and Bioengineering* (BIBE'07), 2007, pp.872-879.
[51] The MotifNetwork website. http://www.motifnetwork.org
[52] C. Vogel, S.A. Teichmann, J. Pereira-Leal, "The Relationship Between Domain Duplication and Recombination", *J. Mol. Biol.*, **346**, 2005, 355-365.
[53] J. Nieplocha, B. Palmer, V. Tpparaju, M. Krishnan, H. Trease and E. Apra, "Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit", International Journal of High Performance Computing Applications, Vol. 20, No. 2, 203-231p, 2006. NXTVAL was originally conceived as part of TCGMSG which is currently distributed with Global Arrays.
[54] W. Gropp, E. Lusk, R. Thakur, Using MPI-2: Advanced Features of the Message-Passing Interface, Scientific and Engineering Computation Series, MIT Press, 1999

Appendix 1

55 Metazoan genomes were processed to generate to the data in Fig 14. The specific names and data used are collected in the following table. The order in the table corresponds the order in Fig 14.

Organism	Number of input sequences	Number of hits
Saccharomyces_cerevisiae	5880	42684
Acyrthosiphon_pisum	10466	113258
Apis_mellifera	11062	126673
Sorex_araneus	13192	191756
Tarsius_syrichta	13561	207975
Anopheles_gambiae	13621	122173
Ciona_intestinalis	13842	151666
Drosophila_virilis	14491	106435
Drosophila_mojavensis	14595	104707
Spermophilus_tridecemlineatus	14830	229102
Drosophila_grimshawi	14986	110808
Drosophila_erecta	15048	107928
Felis_catus	15048	237266
Drosophila_ananassae	15070	108027
Drosophila_simulans	15415	95020
Oryctolagus_cuniculus	15438	226921
Otolemur_garnettii	15448	238684
Tupaia_belangeri	15462	232168
Drosophila_willistoni	15513	108610
Dasypus_novemcinctus	15539	226297
Dipodomys_ordii	15750	243915
Ochotona_princeps	15993	242427
Procavia_capensis	16003	255862
Drosophila_pseudoobscura	16071	120611
Drosophila_yakuba	16082	109710
Myotis_lucifugus	16232	248124
Microcebus_murinus	16319	254075
Tribolium_castaneum	16422	143626
Drosophila_sechellia	16471	109546
Tursiops_truncatus	16493	255735
Gorilla_gorilla	16782	272869
Aedes_aegypti	16789	159495
Drosophila_persimilis	16878	106767
Pteropus_vampyrus	16931	257706
Nasonia_vitripennis	17531	209930
Equus_caballus	18373	272801

Gallus_gallus	18529	212273
Caenorhabditis_japonica	19505	127890
Caiva_porcellus	19774	253696
Monodelphis_domestica	20185	267592
Drosophila_melanogaster	21317	234385
Bos_taurus	22517	286381
Tetraodon_nigroviridis	23118	325562
Oryzias_latipes	24661	305663
Caenorhabditis_brenneri	26048	157244
Danio_rerio	26709	381558
Caenorhabditis_elegans	27258	213290
Caenorhabditis_remanei	31587	198038
Canis_familiaris	33651	503118
Mus_musculus	34966	401509
Rattus_norvegicus	36496	460723
Homo_sapiens	37742	435647
Macaca_mulatta	38001	444936
Takifugu_rubripes	47841	803051
Pan_troglodytes	51947	636137