Fault Tolerance and Recovery for Grid Workflow Systems

Anirban Mandal
Gopi Kandaswamy
Dan Reed

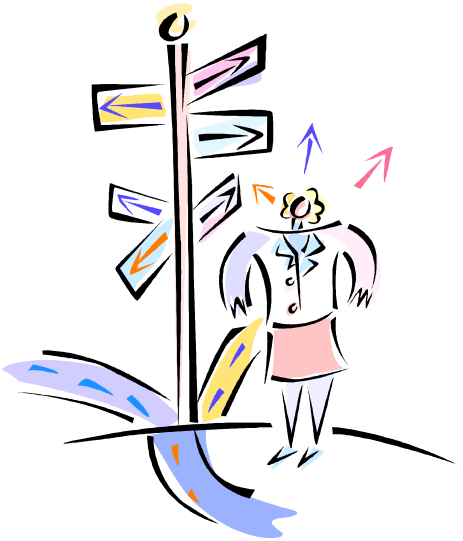
VGrADS Workshop
April 2007

Renci
Renaissance Computing Institute

VGrADS
Virtual Grid Application Development Software Project

Presentation Outline

- Motivation and rationale
- Fault tolerance and recovery service
- Fault tolerance and recovery algorithms
  - over-provisioning, simple restart, migration
- Proposed next steps
- Demonstration
  - LEAD-VGrADS integration



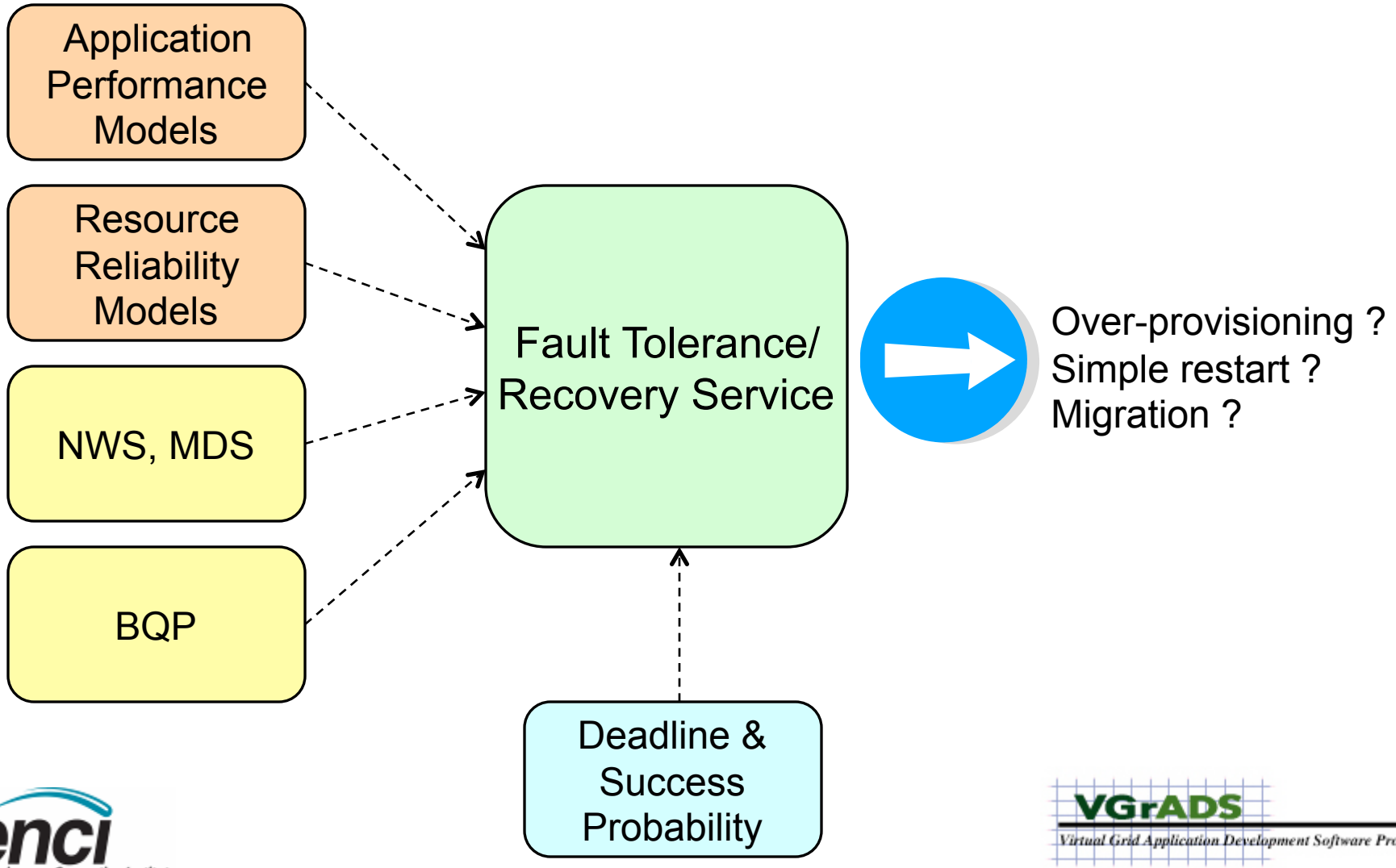
Renci Renaissance Computing Institute

VGrADS Virtual Grid Application Development Software Project

Motivation

- **Reliability and performance are related**
  - **failure is the limiting case of poor performance**
  - **both involve measures of behavior over time**
- **Large, complex workflows are sensitive to failures**
  - **faults are the norm, rather than exceptions**
    - distributed systems, services and resources
  - **completion "guarantees" are problematic**
    - workflow completion is probabilistic in the presence of faults
- **Many time-critical workflows are deadline driven**
  - **severe weather events, disaster response, …**

Fault Tolerance/Recovery (FTR) Service



Simplified Architecture

Portal

BPEL Workflow Engine

Application Performance Models

Resource Reliability Models

NWS, MDS

BQP

Fault Tolerance/ Recovery Service

OVP/ RST/ MIG

Application Service

Notification Service

Deadline & Success Probability

Renci Renaissance Computing Institute

VGrADS Virtual Grid Application Development Software Project

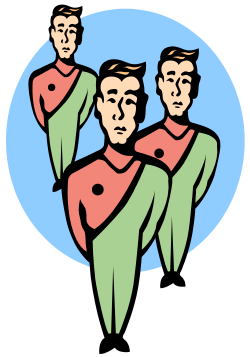
FTR Algorithms

- **Notation**
  - p_i : one hour failure probability of resource *i*
  - h_i : expected execution cost of application on resource *i*
    - queue wait time
    - expected computation time
    - expected communication time
  - *x* : required success probability
  - *d* : required deadline

- **If the reliability function is linear, probability of failure**
  - $m_i = \max(p_i * h_i, 1)$

- **Resource *i* represents (queue, #nodes) combinations**

Over Provisioning



- **Find**
  - **degree and resources for over-provisioning**
- **Number of application copies**
  - **meet a deadline *d* with a success probability *x***
- **Solve the following optimization problem**

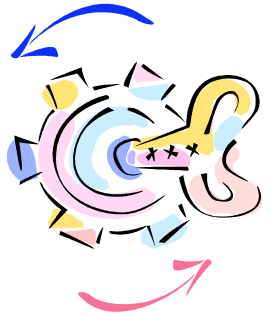
For given *[1..M]* resources, find a partition $P = \{s_1, s_2 \ldots s_n\}$ of *[1..M]* such that

$$1 - m_{s1} * m_{s2} * \ldots m_{sn} \geq x \ \Lambda \ |P| \text{ is minimum } \Lambda \ min\{h_{s1} \ldots h_{sn}\} \leq d$$

Probability of failure

Minimum number of resources meeting deadline

Simple Restart



- Among the current available resources …
  - (universe – faulty resources)
- Find the best resource that
  - meets the deadline and success probability
- Mechanism can also be used for
  - runtime resource selection
- Two FTR mechanisms based on simple restart
  - restart with retries
  - restart if there is no progress

Migration

- **Find the best migration path**
  - **more complex optimization problem**
  - **need to optimize orderings**
- **Solve the following optimization problem**

Find a partition $P = \{s_1, s_2 \ldots s_n\}$ and ordering $\{s_1 > s_2 > \ldots s_n\}$ such that

$\mathbf{1 - m_{s1} * m_{s2} * \ldots m_{sn} \geq x \ \Lambda \ |P| \text{ is minimum } \Lambda \ (t_{s1} + \ldots t_{sn} + q) \leq d,}$

where $t_{si} = (p_{si} * h_{si} > 1)?\ 1/p_{si} : h_{si}$ and q is the migration overhead

Proposed Next Steps

- **Modeling resource reliability**
  - **draw data from job log files in the resources**
  - **maintain constant updates of reliability estimates**
- **Reliability Information Service (viz. BQP)**
  - **"What is the probability that a job, when submitted to a queue at a resource will die before completion because of a failure?"**
- **Balancing multiple mechanisms**
  - **given resource constraints**
    - e.g., over-provisioning and migration
  - **quantifying resource "wastage" with common metric**

Proposed Next Steps (2)

- **Develop runtime rescheduling strategies**
  - **requires following changes in VGrADS software**
    - ability to re-acquire slots
    - ability to relinquish unnecessary slots
    - ability to accommodate FTR directives
    - ability to schedule remaining workflow
  - **requires following change in workflow engine**
    - ability to run a re-scheduled workflow

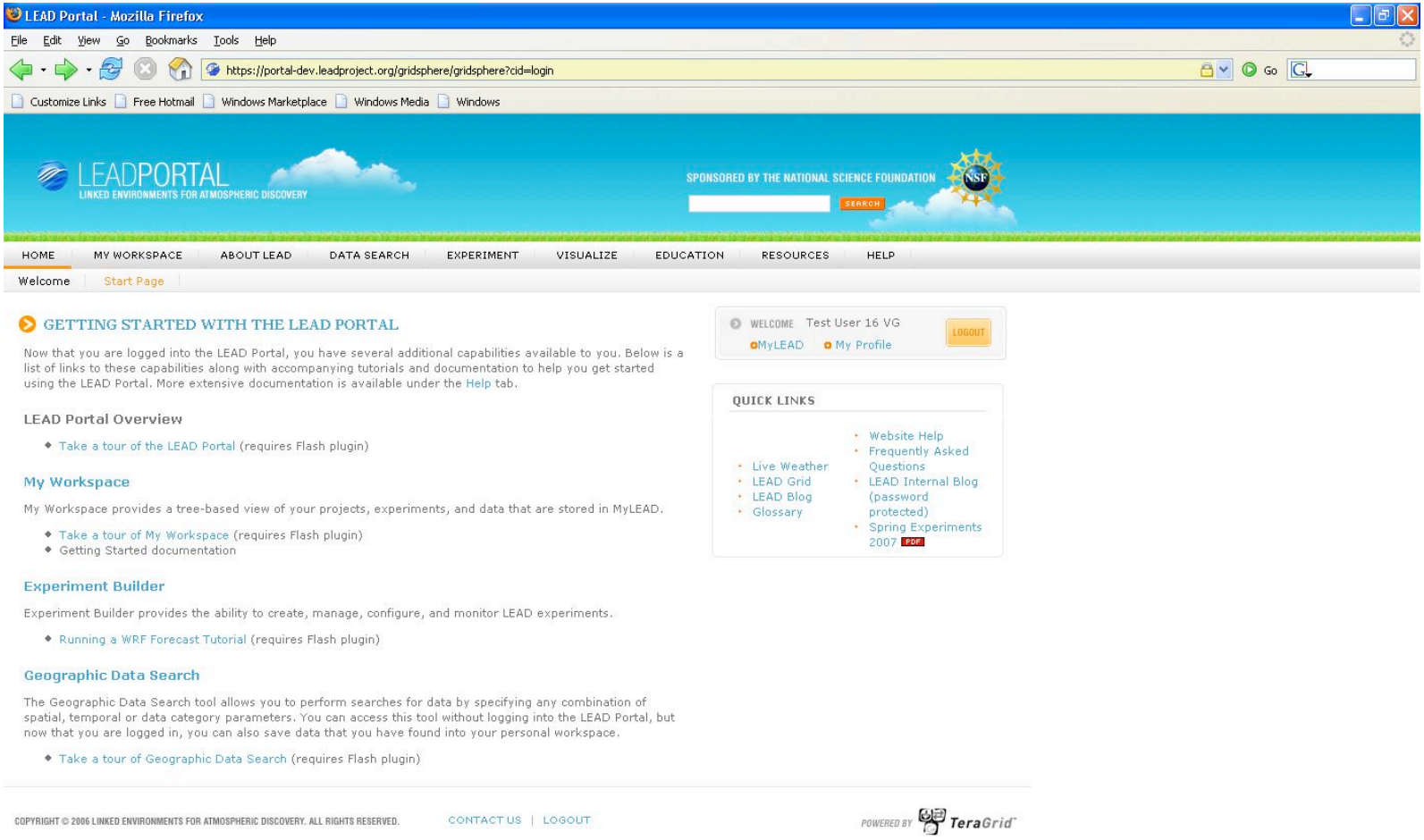
Proposed Next Steps (3)

- **Deadline estimates**
  - **given a deadline for the entire workflow**
    - find deadlines for individual applications
  - **requires critical-path analysis**

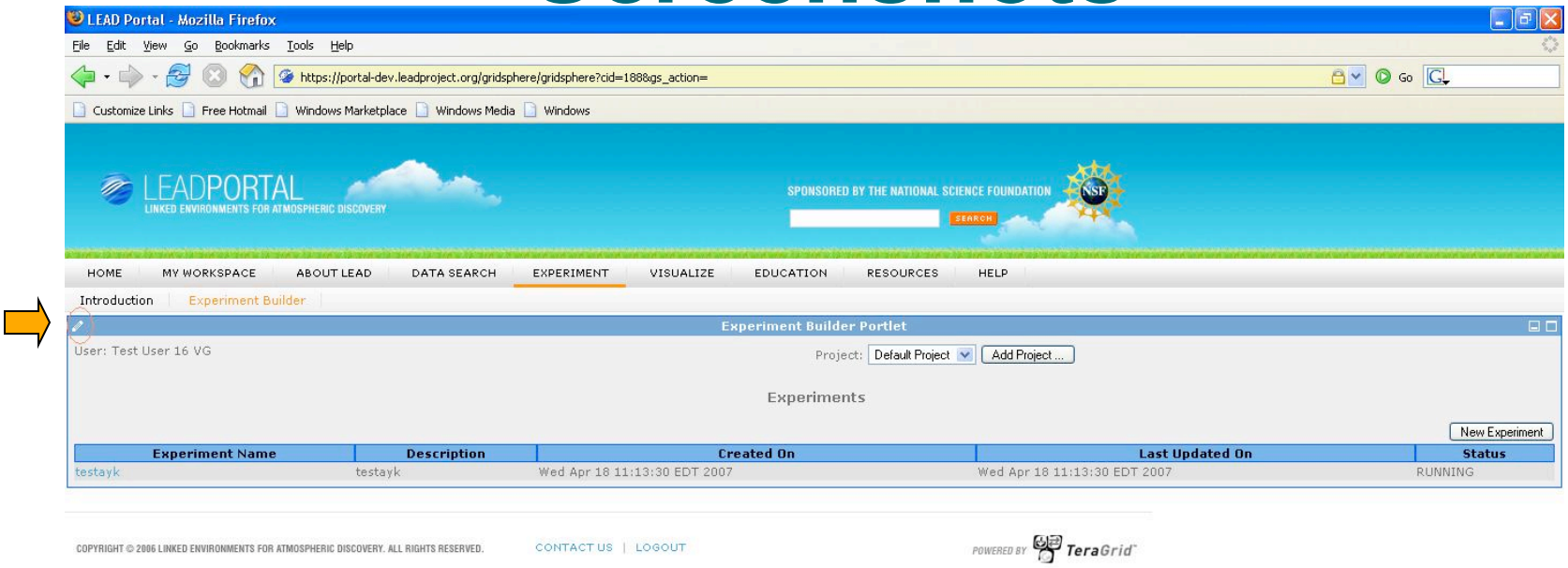
FTR Demonstration

- **Launch workflow from the LEAD portal**
  - build new experiment
  - select forecast region and workflow
  - fetch data
- **FTR service manages application execution by**
  - choosing the fault-tolerance mode (OVP, RST)
  - invoking application service(s) as per current mode
  - monitoring for failures
- **Workflow composer shows workflow progress**
  - status of application currently running
  - workflow execution status via notifications etc.

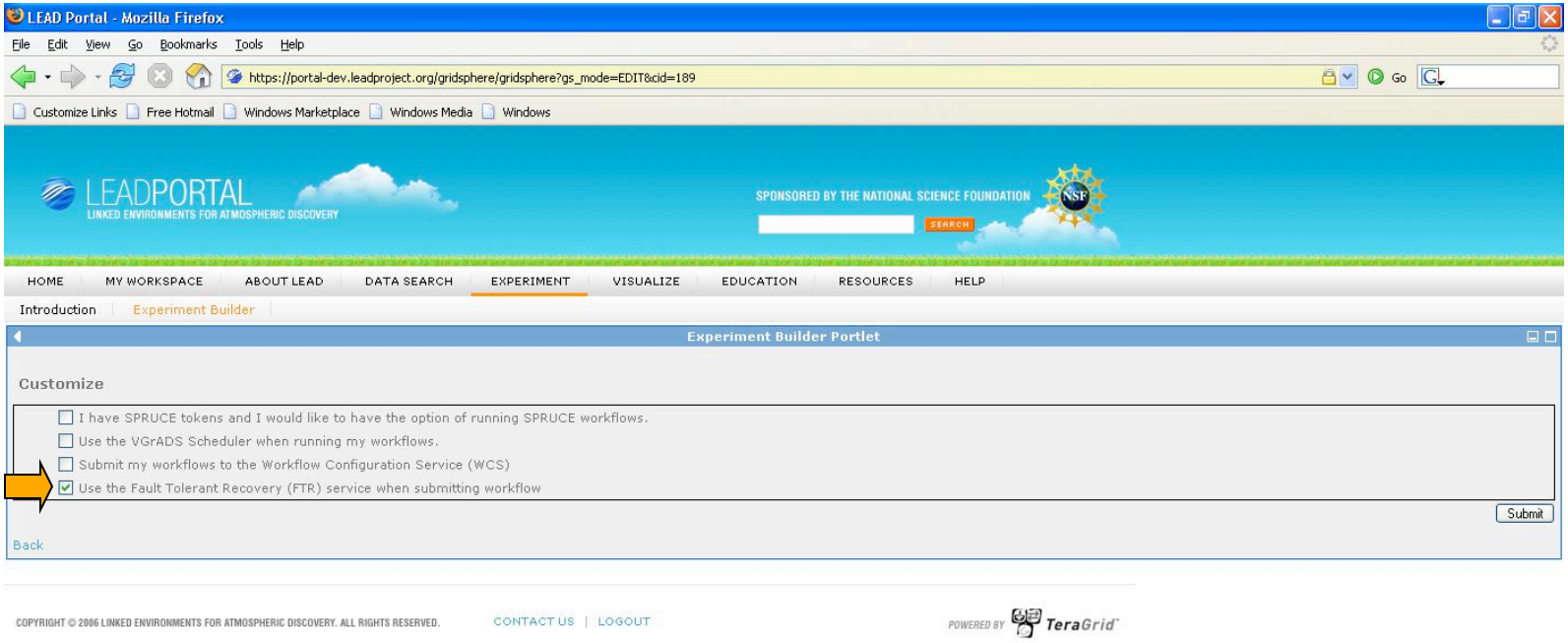
Screenshots



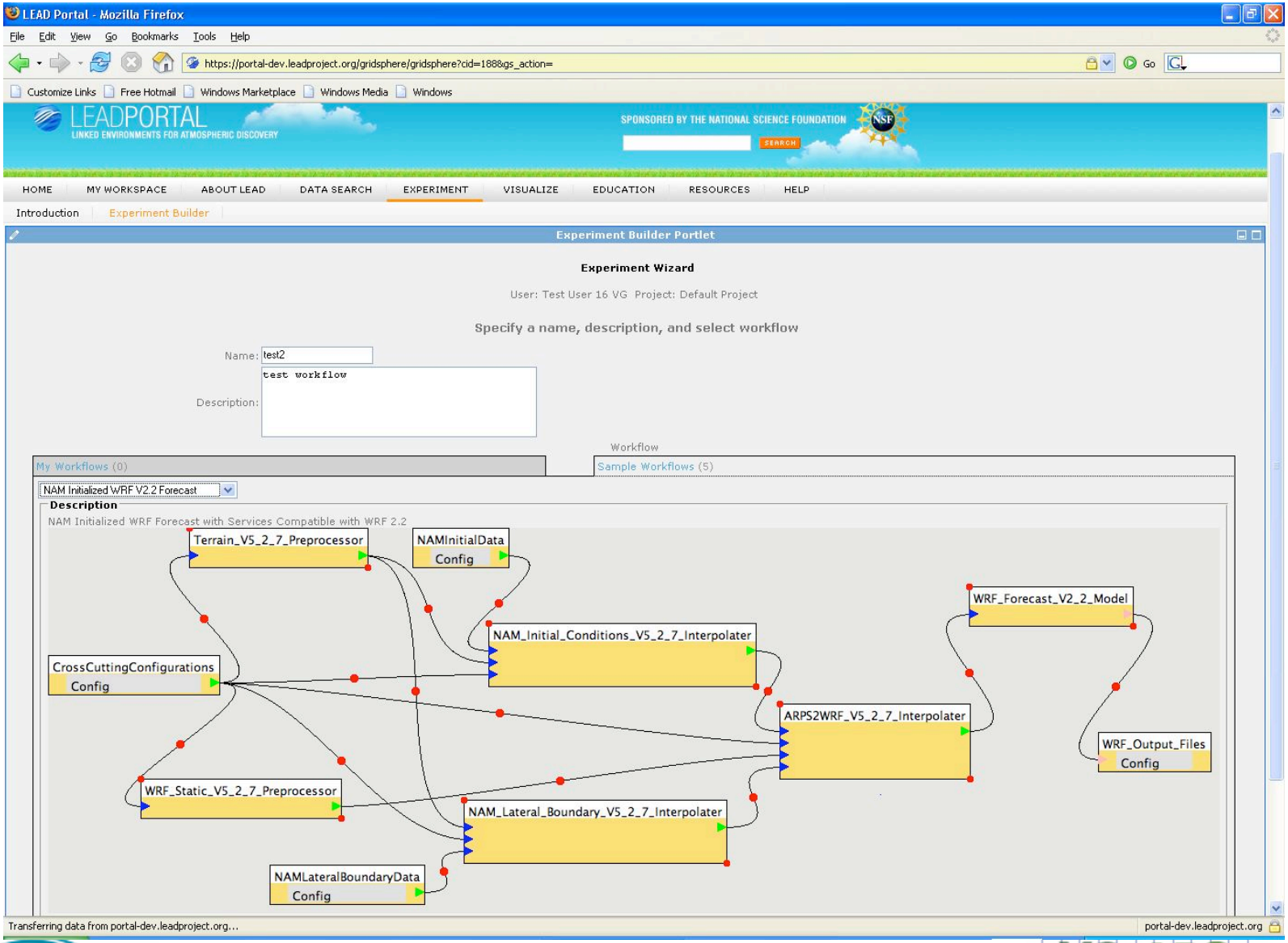
Screenshots



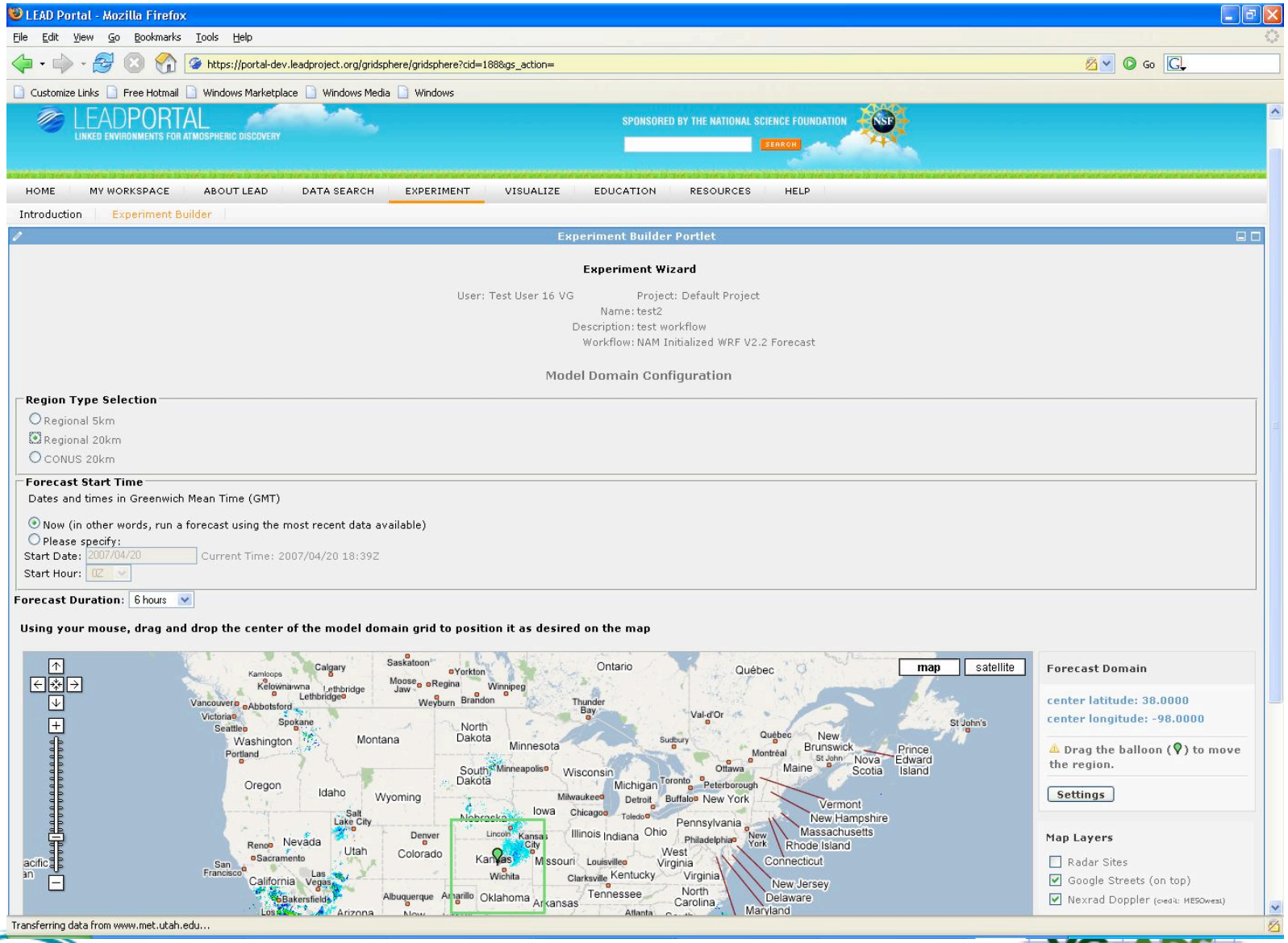
Screenshots



Screenshots



Screenshots



Questions ?