VGrADS Execution System Plans

February 10, 2004

Editor: Andrew A. Chien

with input from Henri Casanova, Rich Wolski, Jack Dongarra, Fran Berman, Dan Reed, Carl Kesselman, Yang-seok Kee, Alex Olugbile, Richard Huang

1. **What is a Virtual Grid?**
   - something that accelerates and improves decision making about resources
     - provides structure for information collection by the execution system
     - provides structure for efficient presentation to the application / libraries / programming system
     - enables the scope of resource monitoring and scheduling to be reduced, improving scalability
     - enables proactive and reactive resource monitoring, acquisition to improve properties of performance, reliability, stability, security, etc.
   - focus is on resource management, selection, allocation, and binding
   - many attributes can be represented, including security, reliability, as well as traditional performance measures
   - improved capability enables better decision making and scaling to larger resource environments
   - introduces two views of resources
     - those currently being used the application/system
     - those being monitored, and selected against

2. How are Virtual Grid abstractions defined?

There are many possibilities.
   - top-down (from the application); prescriptive in describing a desired virtual environment; driving the virtualization; all of the interaction of these then managed transparently in the underlying grid system
   - bottom-up (from the resources and structures); system resources organized into virtual grids; enhanced properties with respect to those grids; rapid selection and binding
   - resource properties
   - communication structure properties
   - and aggregates over these such as reliability, quality of service

3. Oracle Style Example:
     - Local association of a mainframe and collection of desktops
     - Application needs of these associated in a virtual grid, which can then be tied to underlying physical resources
     - If no underlying resource managers couple these resources together (a mainframe and set of workstations), the you might select from a pool of mainframes and from a pool of units of 100's of workstations
     - If the mainframes and workstations are precoupled together, this may lead to overly specialized, inflexible resource organizations

4. A Straw-man Virtual Grid Abstraction

- Application / Program Preparation System Facing View
  - Virtual Grid Abstraction is an abstract description of a set of resources; this is the basis of the performance contract between PPS and Execution System
  - The application exports this view and it is realized in a Virtual Grid to support the abstract parallel machine or abstract component machine or another
  - Examples: uniform symmetrically connected machine, uniform mesh machine, database and cluster, big bag of processors, fully-connected, mesh-connected, pipeline, tree, others.
    - How general a description language do we NEED? Or WANT?
  - Virtual Grid Services: raw compute execution interface, high level specification of virtual grid views, communication primitives, and interfaces to request a new virtual grid;
  - Application Required Services: callbacks for CHECKPOINTING/SYNCH/LOGGING and RESCHEDULING
  - All are exposed as SERVICES so they can be used electively
    - Can we meaningfully support use of the system and modification at multiple levels of abstraction?
- Virtual Grid Runtime View
  - Modules which implement the VG abstractions from an underlying pool of resources that are too large, unreliable, partially accessible, varying characteristics, and varying connectivity
  - Exploit the resource classes and statistical characterization to form complex ensembles of resources which implement the virtual grid view required by the programming system. Certain things are presumed not to matter. In this sense, the resource classes can form another layer of the system – resource and resource structure centric, they can enable the virtual grid realization systems to achieve higher performance and capability – if they are trusted.
  - Built-in performance monitoring to determine the efficacy of the realization of the abstractions, and application callback for CHECKPOINTING and RESCHEDULING or other forms of adaptation to the application should imminent failure or an inability to meet the requirements of the Virtual Grid abstraction view
    - Where do ideas of transparent fault-tolerance fit? We can embed this technology for example in a parallel virtual grid abstraction implementation that provides fault-tolerance?
  - Virtual Grid Runtime Implementation Requires: continuous activity
    - intelligent scoping mechanisms to focus interest and collect relevant dynamic information
    - proactive acquisition of resources to meet current and anticipated needs
    - planning to adapt to anticipated and unanticipated underlying resource environment changes
- Resource Classes
  - Characterization and organization of resources
  - Requires short and long-term monitoring and analysis of resources
  - Many Open Questions
    - What are the meaningful/useful resource classes?
    - How do we both support large-scale of resources, yet refined classification?
    - Is this a multi-classification?
    - Is this centralized or decentralized or both?
- The Virtual Grid / VGrADS runtime lives between the Virtual Grid PPS and Runtime View layers and implements efficient information services, scheduling, and fault-tolerance

These are broad challenges…

5. How do We Make Progress?

- take familiar and important application/workloads and explore issues
  - what type of virtual grid might an application specify
  - how might we exploit these attributes for better selection/scheduling, etc.
  - Initial work on EOL and speeding critical phases
- take typical resource configurations and elicit structure
  - what are the likely configuration of resources in a grid
  - what are their characteristics (static, dynamic)
  - do these naturally fall into structured classification with respect to the needs
  - how might we reduce the scope using a virtual grid mechanism to reduce the number that need be considered
- explore the performance of grid information systems
  - what types of information can be provided with what resolution and accuracy
  - how do these properties scale
  - what techniques are there to make the gathering and distribution of different types of information more efficient and scalable

What might these capabilities look like? How would they be presented?

What might we deliver?

- to the program preparation system team?
- In the immediate term? (next 12 months)
- For the site visit in year 3 (24 months)
- Towards the end of the project (48 months)

What are the implications of the framework?

- how do these affect the interfaces to the program preparation system?
- How to they affect the functionality needed in the program preparation system?

How do they relate to the existing GrADS infrastructure?

- Gradsoft took a single general-purpose view
- VGrads takes a specialist view in presentation to the application
- Underlying elements may be shared
- Abstract performance model – not per application performance models

Year-by-Year Research **Milestones and Tasks**

a. Year 1 Milestones and Tasks:
    i. Execution System/Virtualization:
        - Prototype Resource Virtualization and Abstraction Classes [V1]
        - Virtual Scheduling requirements study [V2]

    ii. Execution System/Performance Provisioning:
        - Initial time-spacereasoning for contracts and signatures [PP1]
    iii. Execution System/Grid Economy:
        - Develop rudimentary simulation of VGrADS resource allocation mechanisms. [GE1]
        - Begin the exploration of Tatonnement, Smale's method, and Continuous-Price Double auctions using simulation. [GE2]
    iv. Execution System/Fault Tolerance:
        - Experimental measurement of Grid & cluster reliability [FT1]
        -

b. Year 2 Milestones and Tasks:
    i. Execution System/Virtualization:
        - Prototype Virtual Grid examples defined [V3]
        - Prototype virtual scheduler [V4]
    ii. Execution System/Performance Provisioning:
        - Extended time-spacereasoning for contracts and signatures [PP2]
    iii. Execution System/Grid Economy:
        - Determine initial pricing conditions and pricing methods that prevent multiple equilibria. [GE3]
        - Verify stability results using simulation environment. [GE4]
    iv. Execution System/Fault Tolerance:
        - Prototype fault tolerant library [FT2]

c. Year 3 Milestones and Tasks:
    i. Execution System/Virtualization:
        - Novel resource selection and virtual scheduling strategy experiments with application kernels on virtual grid environments [V5]
    ii. Execution System/Performance Provisioning:
        - Limited tunable performance/fault-tolerance capabilities [PP3]
    iii. Execution System/Grid Economy:
        - Begin designing experiments to test pricing techniques using VGrADS framework. [GE5]
        - Continue simulation experiments to evaluate resource allocation efficiency. [GE6]
    iv. Execution System/Fault Tolerance:
        - Consider novel techniques [FT3]

d. Year 4 Milestones and Tasks:

i. Execution System/Virtualization:
   - Improved Resource virtualization and virtual scheduling approaches based on experiments with additional virtual grid environments [V6]

ii. Execution System/Performance Provisioning:
   - Extended tunable performance/fault-tolerance capabilities [PP4]

iii. Execution System/Grid Economy:
   - Convert GridSAT to use the pricing-based resource allocation. [GE7]
   - Conduct empirical investigation of pricing scheme and its effect on resource allocation stability using GridSAT as driving application. [GE8]

iv. Execution System/Fault Tolerance:
   - Implement novel techniques (e.g. diskless checkpointing) [FT4]

e. Year 5 Milestones and Tasks:

i. Execution System/Virtualization:
   - Continue to improve Resource virtualization and virtual scheduling approaches based on more application kernel experiments [V7]
   - Integrate Resource Virtualization and Abstraction Classes into VGrADS software [V8]
   - Integrate virtual scheduling strategies into VGrADS software [V9]

ii. Execution System/Performance Provisioning:
   - Limited validation and assessment [PP5]

iii. Execution System/Grid Economy:
   - Design experiment to investigate allocation efficiency under various pricing schemes. [GE9]
   - Target second VGrADS-enabled application (to be determined) as a driving application. [GE10]
   - Verify using both GridSAT and second application. [GE11]

iv. Execution System/Fault Tolerance:
   a. Limited validation and assessment [FT5]