

Virtual Grids

- Last Time
 - » Adaptation by Applications
 - » What do you need to know? To do it well?
 - » Grid Application Development Software (GrADS)
- Today
 - » Virtual Grids
 - » Virtual Grid Application Development Software (VGrADS) Project
- Reminders/Announcements
 - » No Class on Thursday, April 28th

CSE225 – Lecture #8

Today's Readings

- Andrew A. Chien, Henri Casanova, Yang-Suk Kee, Richard Huang, Dionysis Logothetis, and Ken Yocum, The Virtual Grid Description Language: vgDL, UCSD Technical Report CS2005-0817. And Update to The Virtual Grid Description Language: vgDL, Version 0.96, March 16, 2005.
- Yang-Suk Kee, Dionysios Logothetis, Richard Huang, Henri Casanova, and Andrew A. Chien, Efficient Resource Description and High Quality Selection for Virtual Grids, In Proceedings of the IEEE Conference on Cluster Computing and the Grid (CCGrid 2005)
- Yang-Suk Kee, Henri Casanova, and Andrew A. Chien, Realistic Modeling and Synthesis of Resources for Computational Grids, In Proceedings of the ACM Conference on High Performance Computing and Networking, SC2004, Pittsburgh, Pennsylvania, November 2004
- Yang-suk Kee, Henri Casanova, and Andrew A. Chien, Combined Selection and Binding for Competitive Resource Environments, submitted for publication.

CSE225 – Lecture #8

The Virtual Grid and vg Execution System (vgES)

Andrew A. Chien, Henri Casanova, Yang-suk Kee,
Ken Yocum, Richard Huang, Dionysis Logothetis, and
Jerry Chou
CSE, SDSC, and CNS
University of California, San Diego

VGrADS Site Visit

April 28, 2005

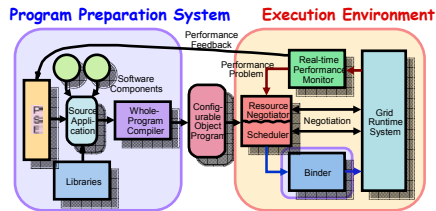


Credits

- **Rice University**
 - Ken Kennedy, Principal Investigator
 - Chuck Koelbel & Mark Mazina, Research Staff
 - Anirban Mandal, Ryan Zhang
- **University of North Carolina**
 - Dan Reed, Principal Investigator
 - Lavanya Ramakrishnan
- **University of Southern California**
 - Carl Kesselman, Principal Investigator
 - Gurmeet Singh
- **University of California, Santa Barbara**
 - Rich Wolski, Principal Investigator
 - Graziano Obertelli
- **University of Tennessee**
 - Jack Dongarra, Principal Investigator
 - Asim YarKhan



Lessons from GrADS



- **Approach Vindicated: Application Driven Adaptation is Crucial**
 - Doing this well is **DIFFICULT**
- **Specifically:**
 - **Implicit Coupling** of Application, Programming Tools, and Runtime requires dealing with complexity at all of these levels simultaneously
 - **Lack of Explicit Resource Abstraction** inhibits expressing and exploiting application domain knowledge for application and resource management
 - **Closed World Selection Model** does not extend to larger, shared resource Grid environments with contended allocation

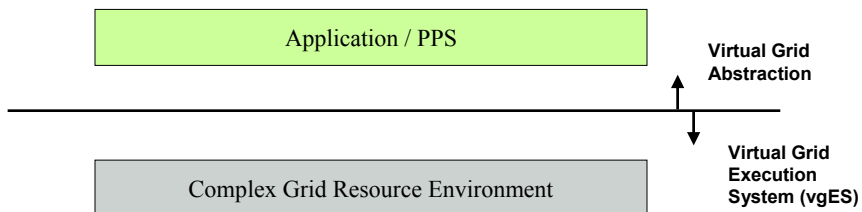
Virtual Grid Research Challenges

- **Separation of Concerns**
 - Application Planning and Management
 - Complex Grid Resource Environment Mgmt
- **Scalable Selection and Binding**
 - Large Resource Pools
 - Competitive, Dynamic Environments
- **Application-Driven Resource Management**
 - Abstraction Level
 - Grid Information
 - Support Fault-Tolerance and Reasoning about Behavior

Virtual Grid Approach

- **Separation of Concerns**
 - “Application Level” Resource Abstraction
 - vgDL: Virtual Grid Description Language
 - Virtual Grid
- **Scalable Selection and Binding**
 - Integrated “Finding and Binding”
 - Overselection and Dynamic Composition
- **Application-Driven Resource Management**
 - VG: Explicit Application Resource Abstraction
 - VG: Unified Resource Information Provider
 - VG: Launch and Monitor Computations
 - VG: Modify to Manage Application Resources

Separation of Concerns: vgDL and VG



- **Virtual Grid Description Language (vgDL)**
 - Applications Describe their resource Needs at Application-level Abstraction
- **Virtual Grid (VG)**
 - Resources Selected, Bound, and Organized into Application-level Abstraction
- **Adaptive Applications: Fault Tolerance and Reasoning about Behavior (Future)**
 - Applications Manage Resources with VG (Application-level Abstraction)
 - Modify the Virtual Grid

Application-Driven Design of vgDL

- **Extensive Application Case Studies (6 months)**
 - EMAN: Single Particle Analysis and Electron Micrograph Analysis
 - Encyclopedia of Life (Bioinformatics)
 - LEAD: Linked Environments for Atmospheric Discovery
 - GridSAT: Boolean Satisfiability Solver (Logic and Test Design)
- **Questions Explored**
 - How do you organize resources?
 - How to you map to them and relation to performance?
 - What is important to control? What is confusing/irrelevant detail?
- **Findings**
 - Small number of Resource Abstractions
 - Many Details Ignored in Application Mapping and Use
- **vgDL Research Hypotheses...**
 - A simple language for Description and "Finding and Binding" is possible
 - Simplicity supports Portability and Robustness for Applications

Virtual Grid Description Language (vgDL)

- **vgDL provides application-level resource abstraction**
 - **Aggregates or Collections**
 - ClusterOf (Homogeneous, Tightly-Coupled)
 - TightBag (Heterogeneous, Tightly-Coupled)
 - LooseBag (Heterogeneous, Loosely-Coupled)
 - **Individual Resource Attributes (extensible)**
 - CPU, Speed, Memory, Disk, Software, Hostname, etc.
 - **Couplers**
 - HighBW, LowBW, Close, Far
- **Preferences**
 - Scalar Ranking Function, Arithmetic on Attributes
- **Advanced and Extent Reservation (start time, stop time)**
- **Quantity of Resources (service units)**

Virtual Grid Description Language (vgDL)

```
Vgrid := VgDefineExpr ["at" Time ]
VgDefineExpr := Identifier "=" VgExpr
Identifier := String
VgExpr := VgSubExpr | VgDefineExpr ("close" | "far" | "highBW" | "lowBW") VgDefineExpr
VgSubExpr := VgAssociatorExpr | VgNodeExpr | "{" VgExpr "}"
VgAssociatorExpr := VgBagExpr | VgClusterExpr
VgBagExpr := ("LooseBagof" | "TightBagof") "(" Identifier ")" "[" MinNode ":" MaxNode "]"
["[" Rank "=" ArithmeticExpr "]" "{" VgDefineExpr "}"]
MinNode := Integer
MaxNode := Integer
Number := Integer
VgClusterExpr := "Clusterof" "(" Identifier ")" "[" MinNode ":" MaxNode "]" [ "[" RedlineExpr "]" ]
["[" Rank "=" ArithmeticExpr "]" "{" VgDefineExpr "}"]
MinTime := Integer
MaxTime := Integer
VgNodeExpr := "[" RedlineExpr "]" ["[" Rank "=" ArithmeticExpr "]" ]
RedlineExpr := CondAndExpr [ "|" CondAndExpr ]* [ "," Predicate ]
CondAndExpr := EqualExpr [ "&&" EqualExpr ]*
EqualExpr := RelationalExpr [ ("=" | "!=") RelationalExpr ]*
RelationalExpr := AddExpr [ ">=" | "<=" | ">" | "<" AddExpr ]*
AddExpr := MultExpr [ ("+" | "-" MultExpr ]*
MultExpr := UnaryExpr [ ("*" | "/" UnaryExpr ]*
UnaryExpr := Integer | Float | Attribute | "(" RedlineExpr ")" |
(("Cluster" | "TightBag" | LooseBag) "." Attribute)
Predicate := "Required" "(" Attribute [ "," Attribute ]* ")"
Attribute := String
ArithmeticExpr := ArithMultExpr [ ("*" | "/" ArithMultExpr ]*
ArithMultExpr := ArithUnaryExpr [ ("*" | "/" ArithUnaryExpr ]*
ArithUnaryExpr := Integer | Float | Attribute | "(" ArithmeticExpr ")" |
(("Cluster" | "TightBag" | LooseBag) "." Attribute)
```

Virtual Grid Application Development Software Project

vgDL Example

```
BagOfClusters=
  LooseBagOf (N) [10:100]
    [Rank=LooseBag.Nodes]
    {N=ClusterOf (M) [8:32]
      {M=[ (Memory)>=1024
        &&(Disk>2048) ]
      [Rank = Clock]}
    }
}
```

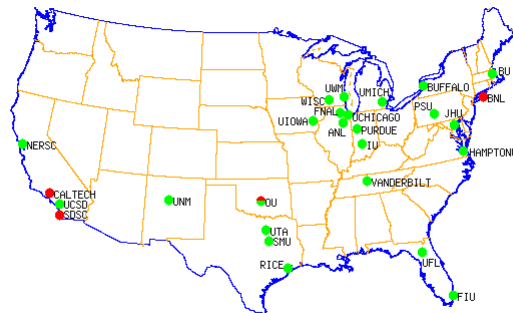
- **Based on EMAN Resource Abstractions**
 - Workflow (loosely coupled)
 - Workflow Nodes are MPI Jobs (clusters)
 - Specific Cluster Node Requirements
- **Simple, Flexible, Lots of Choice**

VGrADS
Virtual Grid Application Development Software Project

Scalable Selection and Binding

- **Traditional Model**
 - Selection based on Static Resource Attributes
 - Plan Application Execution
 - Bind Resources
 - Execute Application
- Works in a Private Resource Environment: “What I want, I get”
- Doesn't work in a competitive Grid resource environment: “What I want, everyone else wants too!”
- Problem:
 - Selection Ignores Availability of or Demand for Resources
 - Binding May not Succeed
 - Application May not do well
- Current solutions: Queue and Wait

Example: iVGD Resource Management

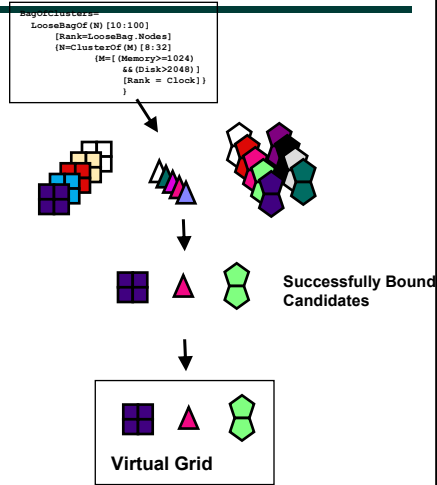


Wed Apr 13 22:50:09 GMT 2005

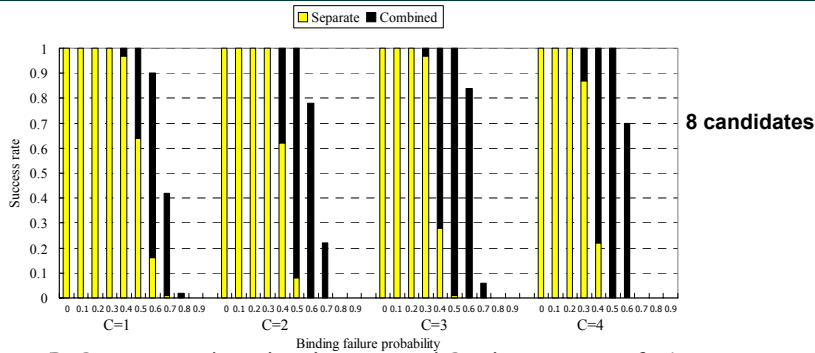
- **VDT: Chimera and Pegasus construct Workflow plan (selection)**
 - Selects resource pools (resource managers) for each workflow node
 - Local Resource Managers (e.g. Condor) determine when they run!
- Problem: Can't control performance with VDT alone!
- **VG Approach: “Find and Bind” Resources, enabling Application Control of Scheduling on Bound Resources (2-level)**

vgFAB: "Finding and Binding"

- Use vgDL (and rank) to enumerate a number of candidates for each part of request
- Attempt to bind candidates for each part based on vgDL ranking
- Compose successfully bound parts into a VG and returns to application
- If didn't succeed for all parts, try iteratively with more candidates or fail

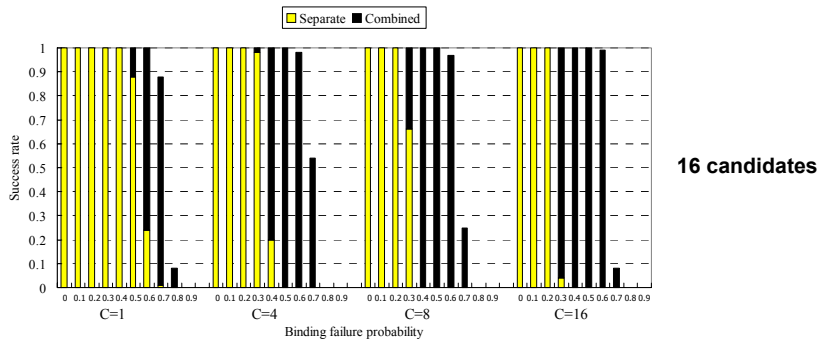


Finding and Binding in Competitive Resource Environments



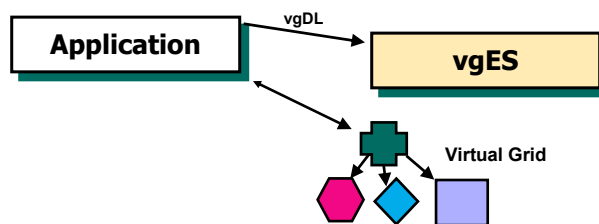
- **vgFAB uses Combined Selection and Binding to Satisfy Large, Complex Requests in Competitive Resource Environments**
 - Combined Approach better in all cases
 - Combined Approach Much better than Separate Selection in Competitive Environments
- **Conjecture: This may make Synchronous Use of Grid Resources in Competitive Resource Environments Possible**

Finding and Binding in Competitive Resource Environments (cont.)



- As Request Complexity Increases, Combined Selection and Binding can tolerate high resource utilizations
 - Enables Synchronous Use of Resources
- As much as doubles the Binding Failure Rate which can be tolerated
 - Separate: 30% for 8 and 16 component descriptions
 - Combined: 60-70% for 8 and 16 component descriptions

Application-Driven Resource Management: the Virtual Grid (VG)



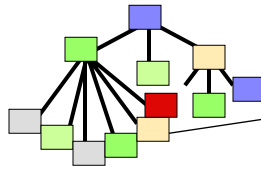
- Life-cycle of a Virtual Grid
 - Application sends vgES a vgDL request
 - vgES (vgFAB) creates VG and returns to Application
 - Application Uses VG (runs jobs, reads resource attributes, gets notifications from monitors, adapts VG, eventually done)
 - Application terminates VG

Virtual Grid = Realized vgDL Request

```

BagOfClusters=
  LooseBagOf (N) [10:100]
    [Rank=LooseBag.Nodes]
    (N=ClusterOf (M) [8:32]
      {M=[ (Memory>=1024)
            && (Disk>2048) ]
          [Rank = Clock]}
      )
  
```

vgDL Request



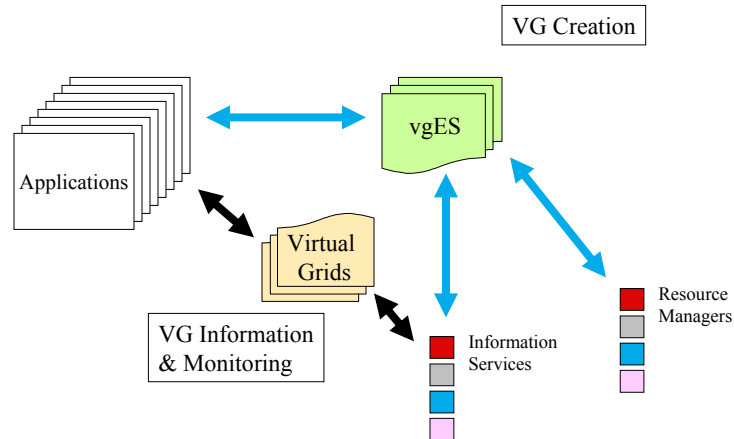
Virtual Grid (VG)

```

Rname: foo.ucsd.edu
Load: 0.6
Pred Load: 2.0
...
  
```

- **vgDL Request to vgES Creates the Virtual Grid (VG)**
- **Virtual Grid (VG) is an Explicit, Active Entity**
 - Bound Resources and their relation to the Application's vgDL
- **VG Nodes Attributes present Resource Information**
 - Static Information (proc type, speed, location, etc.)
 - Dynamic Information (load, mem, uptime, prediction, NWS, Ganglia, etc.)
 - Characterization / Classification Information

Virtual Grids (VG)

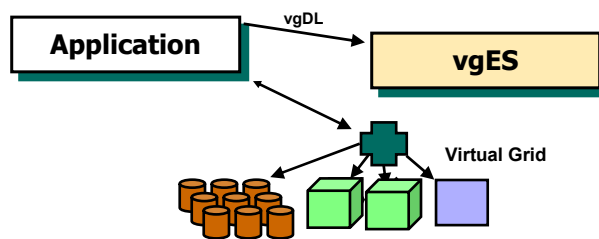


- **Big Picture: Many Applications, vgES Instances, Virtual Grids**

Application-Driven Resource Management: Modifying a Virtual Grid

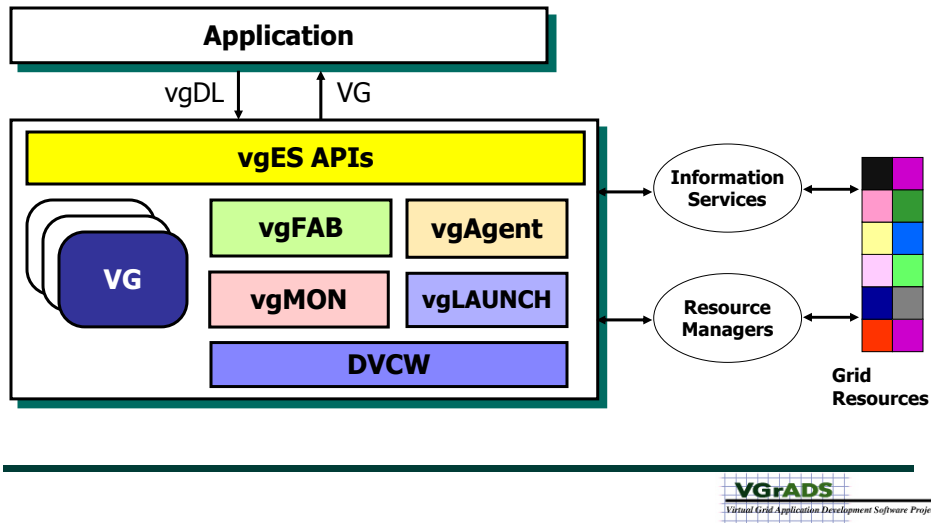
- **Application Reasons to Manage Resources**
 - Application of Resource Performance changes (Reschedule)
 - Better or Changed Prediction of Application Needs
 - Degradation of Resources, Availability of Additional Resources
 - Application or Resource Fault-tolerance Needs (Reconfigure)
 - Change in Predicted Availability, Failure Rates
 - Change in Application Vulnerability and Needs
- **Change the Virtual Grid (Future)**
 - Add Resources: `VG::addNode()`
 - Graft into the VG Structure where desired
 - Release Resources: `VG::removeNode()`
 - Remove from VG structure and release
 - Snapshot for Future Runs: `VG::getDesc()`
 - Produces vgDL which will create similar configurations

Examples of Virtual Grid Evolution



- **Application uses vgDL and vgES to Create a Virtual Grid**
 - Fault Tolerance: Detect and Replace Resources which Fail Partially, or are Re-characterized with Undesirable FITS rates; or Augment
 - LEAD: Add to Virtual Grid as Needed to Meet Evolving Needs of Real-time Prediction and Response

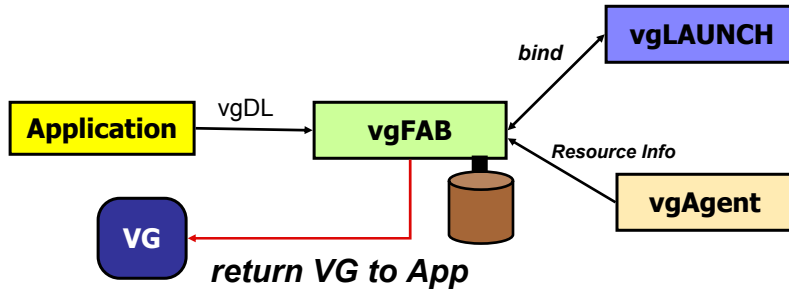
System Architecture: vgES implements Virtual Grid



vgES Components

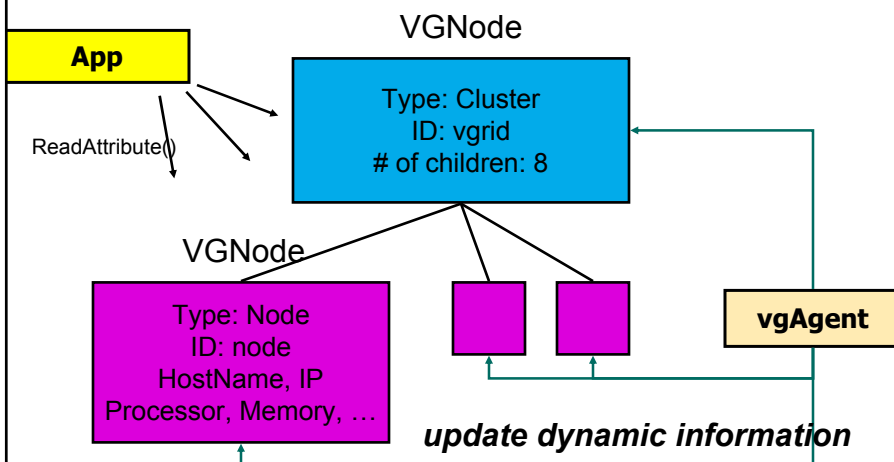
- **vgFAB**
 - A “finder and binder” that performs integrated resource selection and binding
- **vgDL**
 - Virtual Grid Description Language: how an application expresses its resource needs and resource abstractions
- **vgLaunch + DVCW**
 - An application launcher that initiates the application on the bound resources and interfaces to Globus
- **vgAgent**
 - A component that retrieves static/dynamic resource information from existing information services systems
- **vgMON**
 - A distributed monitoring component that ensures resource performance expectations

vgFAB Architecture



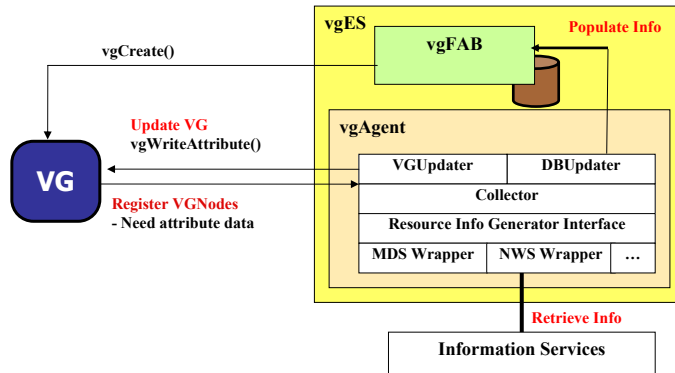
- **vgAgent** provides resource data for finding and binding
- **vgLAUNCH** uses *Globus* to access *Grid* resources

Virtual Grid (VG)



vgAgent

- Retrieve Resource Information from Information Services
- Populate the vgFAB Information store, supports resource selection and binding
- Implement VG Resource Attributes



Achievements to Date (Year 1.5)

- Application Studies to understand Application Resource Abstractions
- Design and Implementation of vgDL Language (Application-level Resource Abstraction)
- Design and Implementation of Integrated "Finding and Binding" Algorithms
- Design and Implementation of Synthetic Resource Generator for Grids (Size, Time, etc.)
- Simulation Experiments for "Finding and Binding" Effectiveness under Various Resource Environments
- Design and Implementation of a Research Infrastructure (vgES 0.7, March 2005) which enables
 - Modular Exploration of Research Issues
 - Experimentation by Large Applications
- And incidentally...
 - Leverages and Integrates with Globus/MDS/Production Grid Resource Infrastructures

Research Activities and Goals

- **Understanding and Effectiveness of vgDL**
 - Experimentation with base vgDL
 - Experimentation with full vgDL (reservations, resource quantities)
 - Evaluation with a range of Applications
 - Large-scale Experiments
- **Evaluation and Understanding of Core vgES**
 - vgFAB Finding and Binding in Competitive Resource Environments (simulation and real experiments)
 - Distributed vgFAB - scaling to even larger systems
 - Efficient presentation of individual and "inter-resource" attributes in VG's
 - Evaluation with a range of Applications
 - Large-scale Experiments
 - Experimentation with use by other Systems (VDT: Chimera and Pegasus)

Research Challenges (cont.)

- **Automatic and Customized Monitoring**
 - vgMON and "separation of concerns"
 - Default and customizable expectations
 - Efficient compilation/implementation of custom monitors
- **Dynamic Virtual Grids**
 - Finding and Binding: Relative to Existing VG
 - Coupling to Fault Tolerance Management
 - Coupling to Reasoning about Behavior
 - Abstraction of vgDL descriptions from Dynamic VG's

Summary

- Virtual Grids attempt to Simplify the Coupling of Applications and Runtime System for Intelligent Adaptation
- Key Strategies are
 - » Separation of Concerns: vgDL and VG
 - » Scalable Selection and Binding: Overselection and Composition
 - » Application Resource Adaptation: VG Attributes and Operations
- Initial Results are Promising...