
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

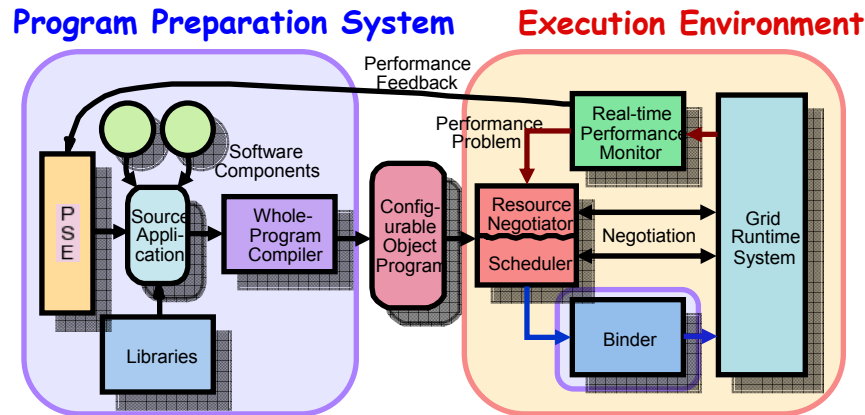
April 28, 2005



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

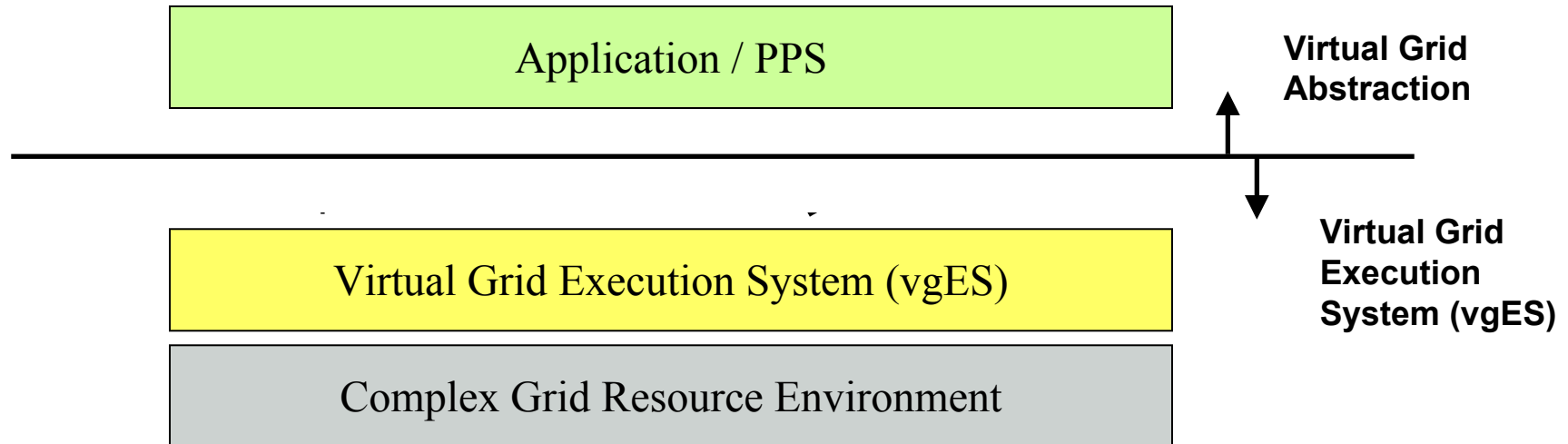


- Approach Vindicated: Application Driven Adaptation is Crucial
 - Doing this well is DIFFICULT
- Specifically:
 - Implicit Coupling of Application, Programming Tools, and Runtime requires dealing with complexity of all levels simultaneously
 - Lack of Explicit Resource Abstraction inhibits expressing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

Virtual Grid Approach

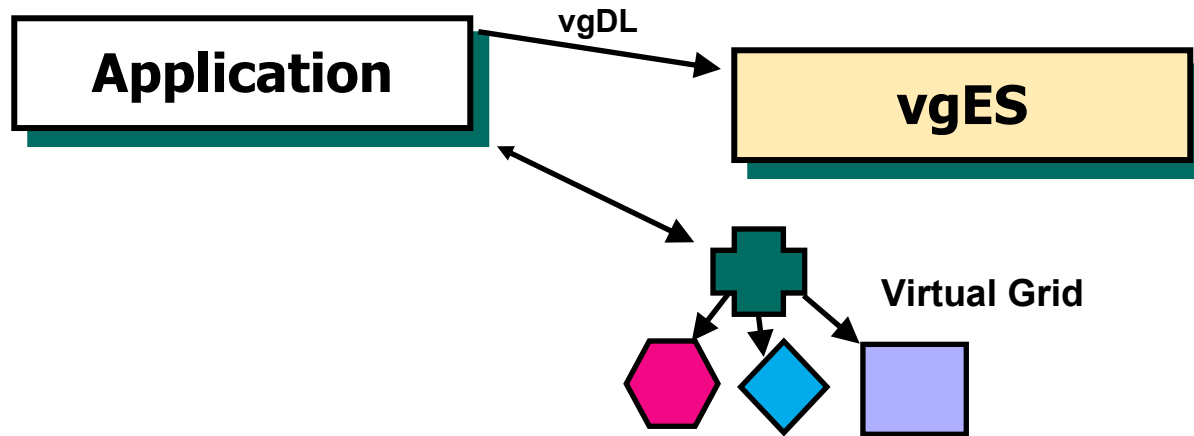
- **Separation of Concerns**
 - Application Planning and Management
 - Complex Grid Resource Environment Management
 - => vgDL and Virtual Grid
- **Scalable Selection and Binding**
 - Large Resource Pools
 - Competitive, Dynamic Environments
 - => Finding and Binding
- **Application-Driven Resource Management**
 - Application-level Abstraction
 - Grid Information
 - => Virtual Grid Explicit Resource Abstraction

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 (Future)**
 - Applications Manage Resources with VG
 - Modify the Virtual Grid

Life Cycle of Virtual Grids



- 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

Application-Driven Design of vgDL

- **Extensive Grid Application 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 do you map your application?
 - How do you reason about performance?
 - What is important to control? What is confusing/irrelevant detail?
- **Findings**
 - Small number of Resource Abstractions
 - Application Mapping and Resource Ignores Detail
- **vgDL Research Hypotheses...**
 - A Simple Application-level Description is better for Applications!
 - Simplicity supports Description Portability and Robustness
 - Loose Specification enables "Finding and Binding"

Virtual Grid Description Language (vgDL)

- **vgDL provides application-level resource abstraction**
 - **Aggregates**
 - **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, Close, Far, LowBW**
- **Preferences**
 - **Scalar Rank Function, Arithmetic on Attributes**
- **Advanced Reservation (Start) and Extent Reservation (Start, Stop)**
- **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 "]"
[ "[" RedlineExpr "]" ] [ " 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)
```

vgDL Example

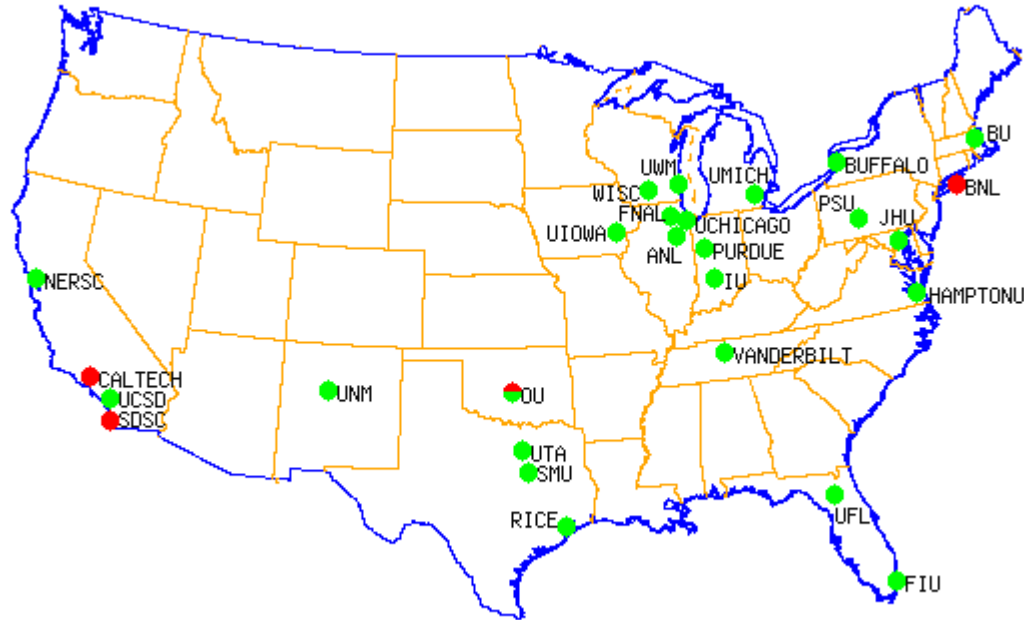
```
BagOfClusters=  
  LooseBagOf (N) [10:100]  
    [Rank=LooseBag.Nodes]  
    {N=ClusterOf (M) [8:32]  
      {M=[ (Memory>=1024)  
          &&(Disk>2048)]  
      [Rank = Clock]}  
    }
```

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

Scalable Selection and Binding

- **Traditional Model: Separate Selection**
 - Select 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 Resource Competition
 - Binding May not Succeed
 - Application May not do well (performance, etc.)
- Current practice: Queue and Wait

Example: Grid3 Resource Management

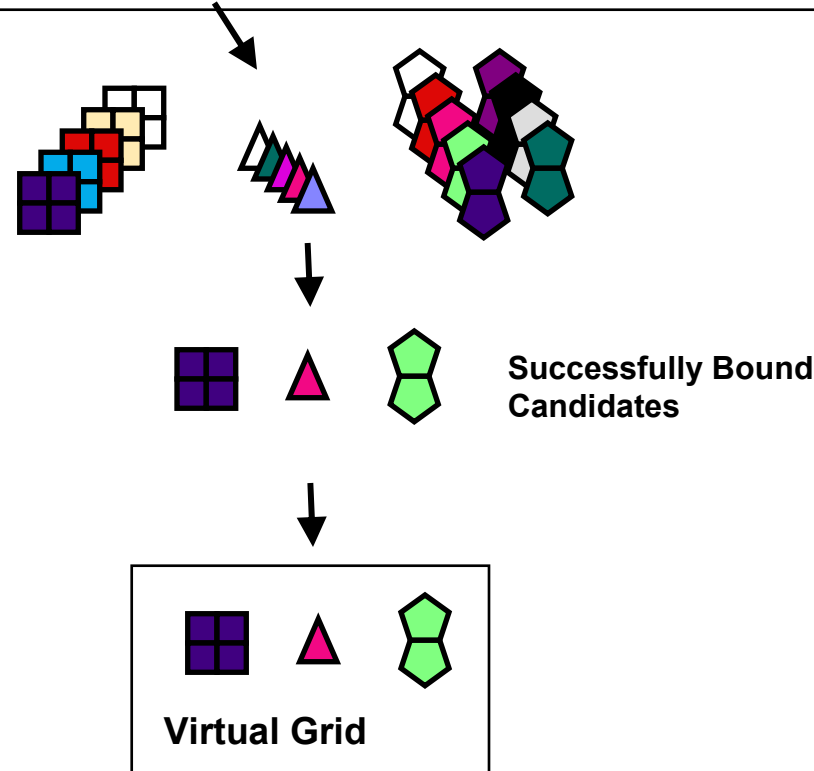


- **VDS: Chimera, Pegasus, Condor, Globus System**
 - Chimera and Pegasus construct Workflow plan (selection), including resource pool for each workflow node
 - Local Resource Managers determine when they run! (binding)
- **Problem: Can't control performance with VDS alone!**
- **VG Approach: "Finding and Binding"**
 - Application Controls Scheduling on VG Resources (2-level)
 - Application Controls Performance

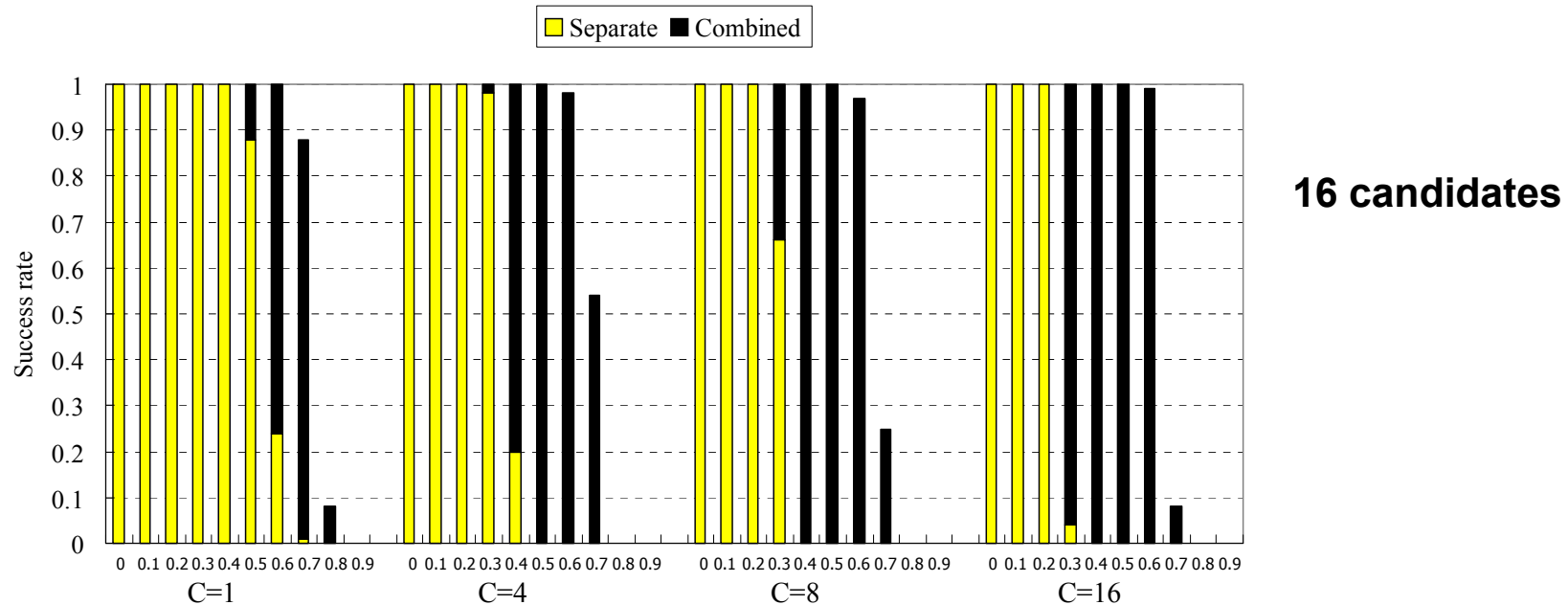
vgFAB: “Finding and Binding”

```
gi = { rsc1 = LooseBagOf(c1)[2:4] {  
    c1 = ClusterOf(node1)[4:8] {  
        node1 = [ (Processor == Pentium4) && (Memory>=4096)] }  
    FAR rsc2 = LooseBagOf(tbl1)[2:4] {  
        tbl1 = TightBagOf(node2)[4:8] { node2 = [ Clock>=2.048 ] } }  
    FAR rsc3 = LooseBagOf(c2)[2:4] {  
        c2 = ClusterOf(node3)[4:8] {  
            node3 = [ Processor == Pentium4 ] [ Rank = Memory ] }  
    }  
}
```

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



vgFAB enables Synchronous Resource Use in Competitive Environments



- **FAB Scalable and Satisfies Complex Requests in Competitive Resource Environments**
 - Combined Better in All Cases; Much Better in Competitive Environments
- **Tolerates Double Binding Failure Rates => Makes Synchronous Use Practical**
 - Separate: 30% for 8 and 16 component descriptions
 - Combined: 60-70% for 8 and 16 component descriptions

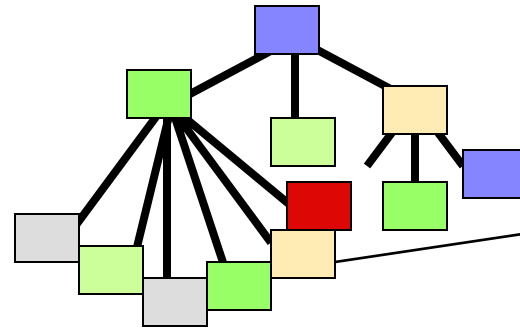
Application-Driven Resource Management: the Virtual Grid (VG)

- **Why Manage Resources?**
 - Application or Resource Performance changes (Reschedule)
 - Application or Resource Fault-tolerance Needs (Reconfigure)
- **Virtual Grid Provides an Application-Level Abstraction for Management**
 - Relates to vgDL
 - Corresponds to Application-level View
 - Single Interface for Resource Information
 - Operations for Computation Launching and Monitoring
 - Modify Virtual Grid; Augment, Remove Elements

Virtual Grid: Explicit Resource Abstraction

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

vgDL Request

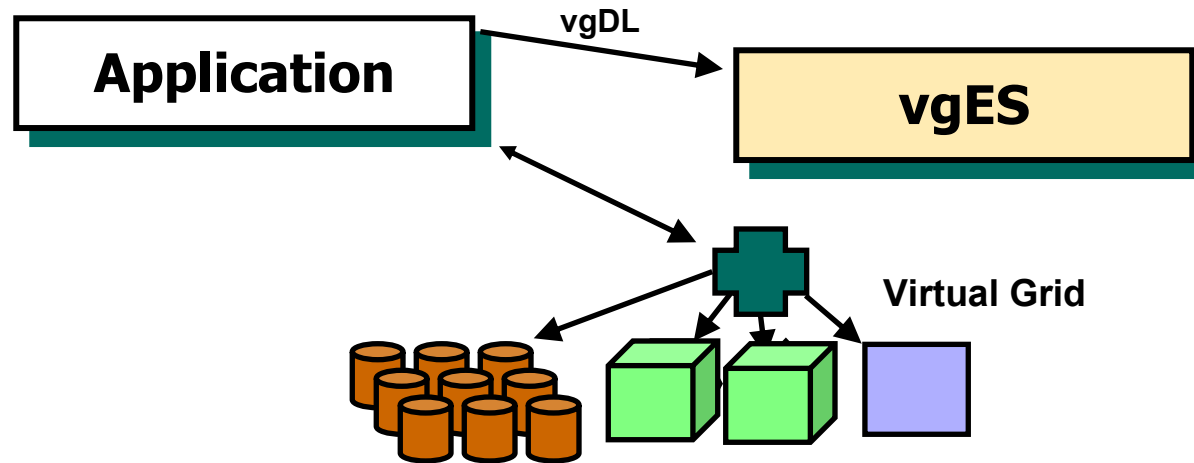


Virtual Grid (VG)

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

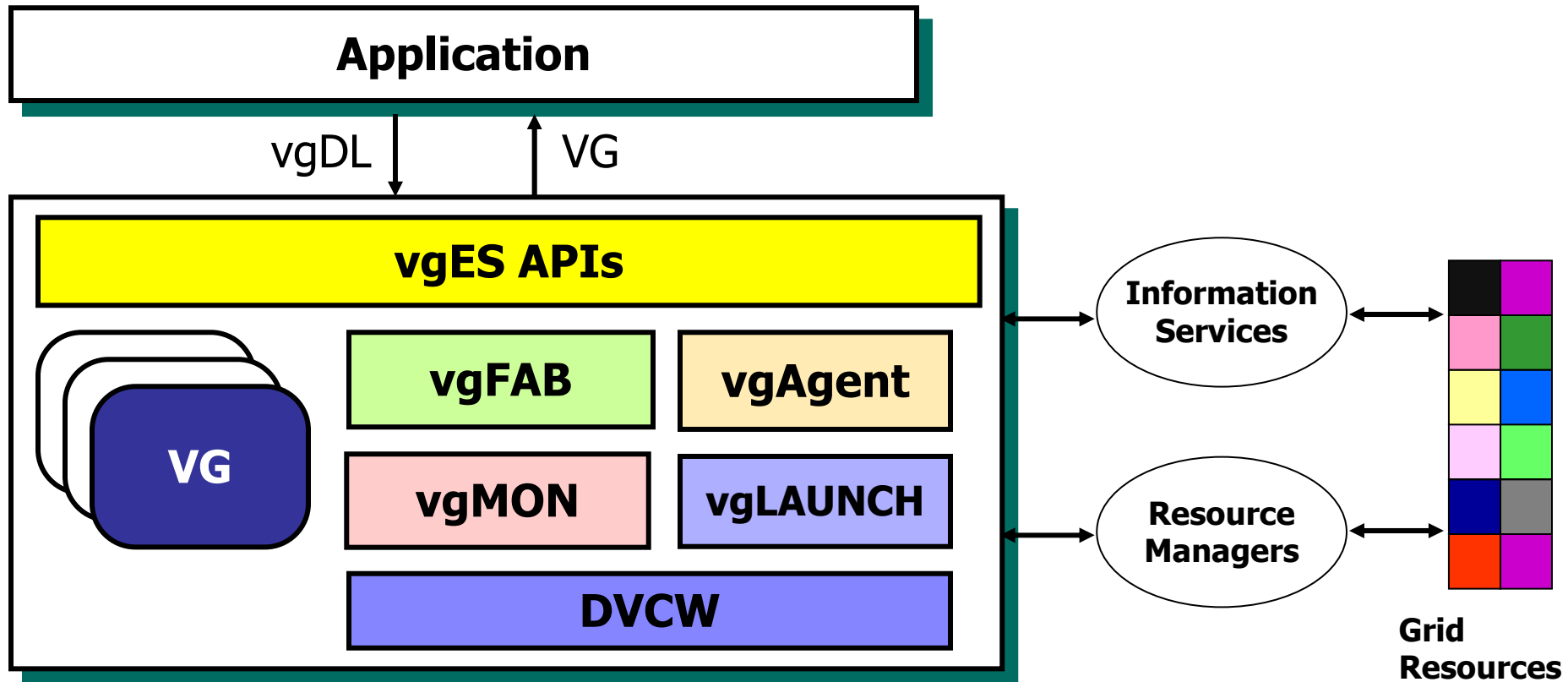
- **vgDL Request Creates Virtual Grid (VG)**
 - Virtual Grid (VG) is an Explicit Entity
 - Each VG Resource corresponds to part of the Application's vgDL
- **VG Nodes Attributes present Resource Information**
 - Static Information (proc type, speed, location, etc.)
 - Dynamic Information (load, mem, uptime, prediction, NWS, Ganglia, etc.)
 - Characterization / Classification Information

Virtual Grid Creation and Evolution



- Life-cycle of a Virtual Grid
 - Application sends vgDL request
 - vgES creates VG and returns it to Application
 - Application Uses VG; Terminates VG
- Application Modifies Virtual Grid
 - Detect and Replace Resources which Fail (FT) or Augment
 - Add to Virtual Grid as Needed to Meet Evolving Needs (LEAD)

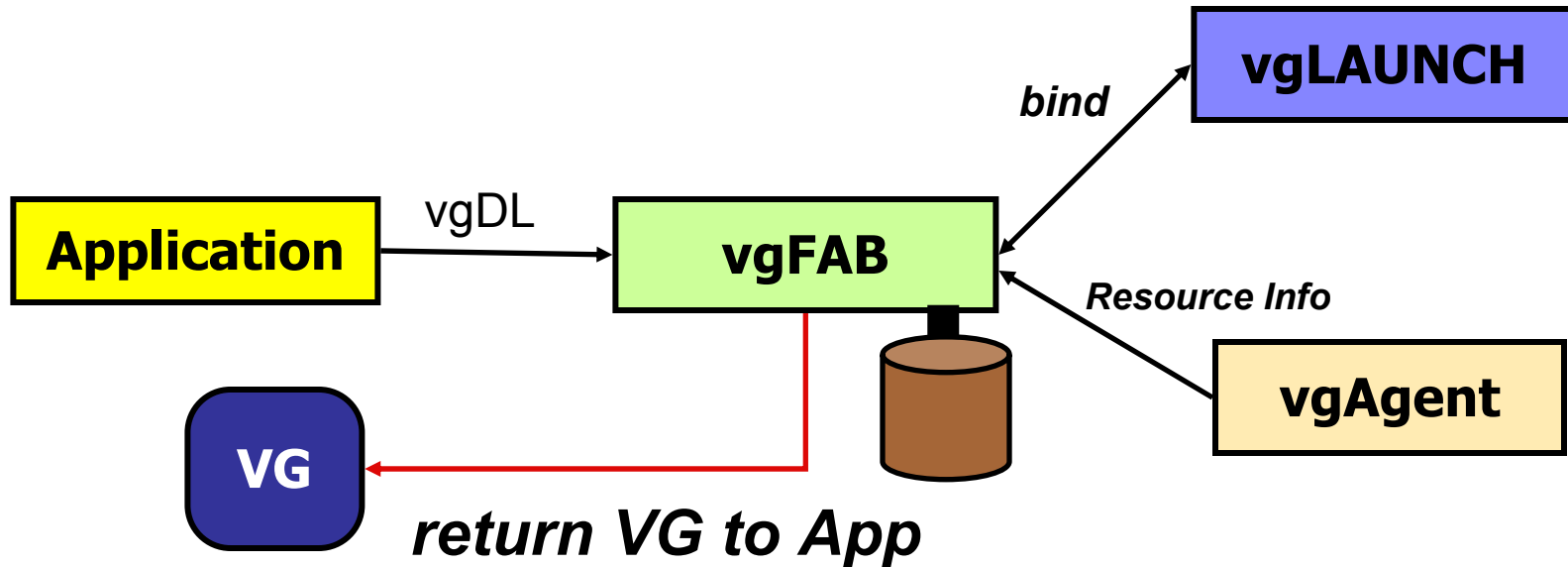
System Architecture: vgES implements Virtual Grid



vgES Components

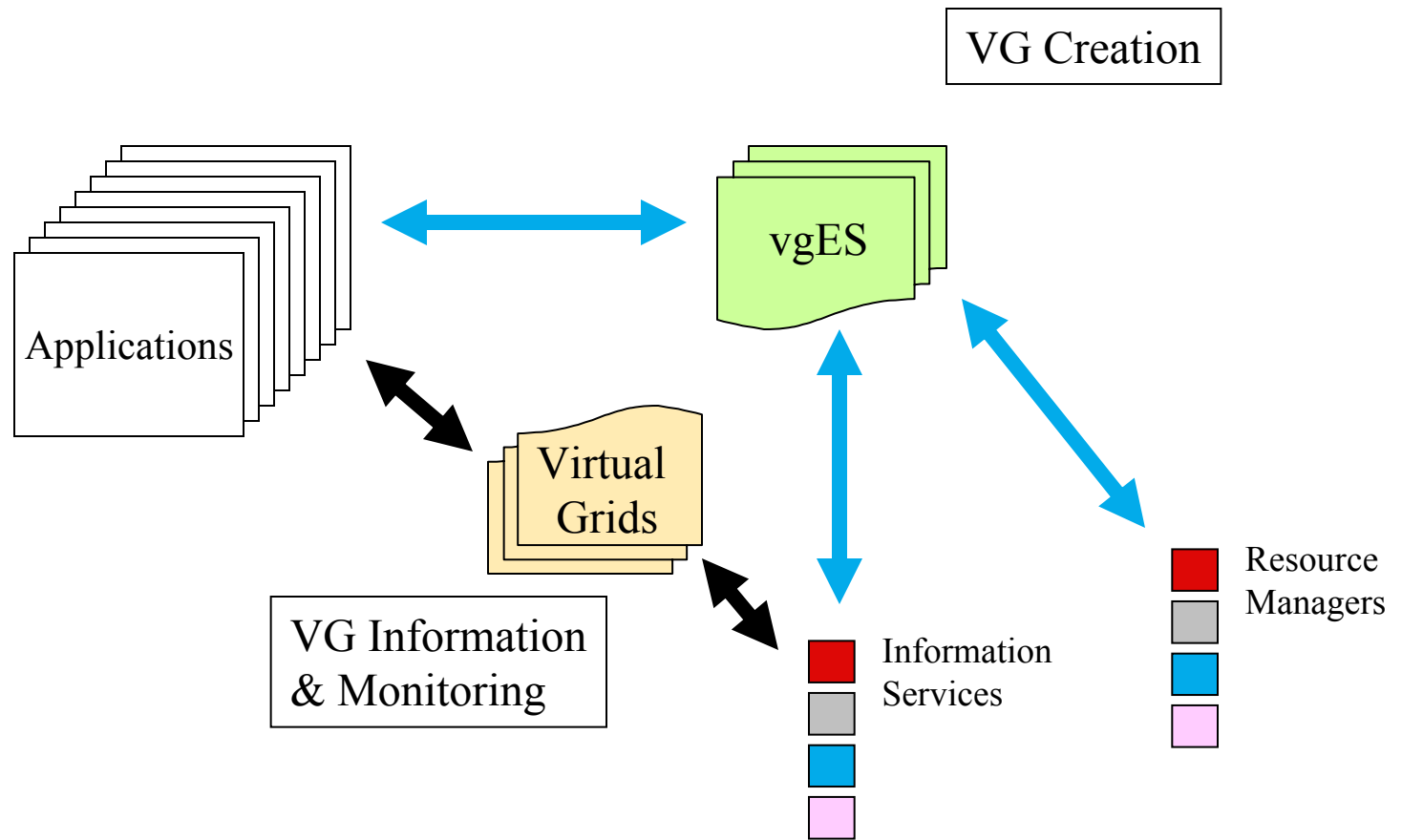
- **vgFAB**
 - A “finder and binder” that performs integrated resource selection and binding
- **vgDL**
 - Virtual Grid Description Language: how an application expresses its resource needs and resource abstractions
- **vgLaunch + DVCW**
 - An application launcher that initiates the application on the 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



- vgFAB accepts vgDL requests; creates Virtual Grids
- vgAgent provides resource data for finding and binding
—Implements VG Attributes
- vgLAUNCH uses DVC/Globus to access Grid resources

Virtual Grids (VG)



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

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
- Simulation Experiments for "Finding and Binding" Effectiveness under Various Resource Environments
- Design and Implementation of Synthetic Resource Generator for Grids (Size, Time, etc.)
- Design and Implementation of a Research Infrastructure (**vgES 0.7, March 2005**) which
 - Realizes the Key VG Ideas
 - Enables Modular Exploration of Research Issues
 - Enables Experimentation with Large Applications
 - Leverages and Integrates with Globus/MDS/Production Grid Resource Infrastructures

Future 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 (EMAN, GridSAT, LEAD)
 - Large-scale Experiments (VGrADS Testbed, iVDGL, TeraGrid,...)
 - Experimentation with use by other Systems (VDT: Chimera and Pegasus)
- **Exploring the Virtual Grid -- 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 & "inter-resource" attributes
 - Evaluation with a range of Applications (EMAN, GridSAT, LEAD)
 - Large-scale Experiments (VGrADS Testbed, iVDGL, TeraGrid,...)
 - Experimentation with use by other Systems (VDT: Chimera and Pegasus)

Future Research Activities and Goals

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

For More Information

- 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.

Morning Break