# CGrADS Program Execution Environment

**Fran Berman**
Director, SDSC and NPACI
Professor, CSE Department, UCSD

http://hipersoft.cs.rice.edu/stc_site_visit/talks/PES.ppt

# Program Execution Environment

- Goal: to provide an execution environment that automatically adapts the application to the dynamically changing resources of the Grid

- Key components
  - —Resource discovery
  - —Scheduling of application on Grid resources
  - —Program launch (on selected resources)
  - —Performance monitoring
  - —Discovery of performance problems and rescheduling

# Why is Grid Program Execution Hard?

- Resource performance is dynamic and hard to model accurately
  - Exact models difficult to develop
  - Resources are shared (contention, unpredictable performance)
- Application performance is also difficult to model
  - Behavior and performance based on environment
- Trade-off between good model accuracy and low execution overhead
- "Chicken and Egg" problem between program preparation system and program execution environment in determining and automating performance-efficient allocation

# Foundations

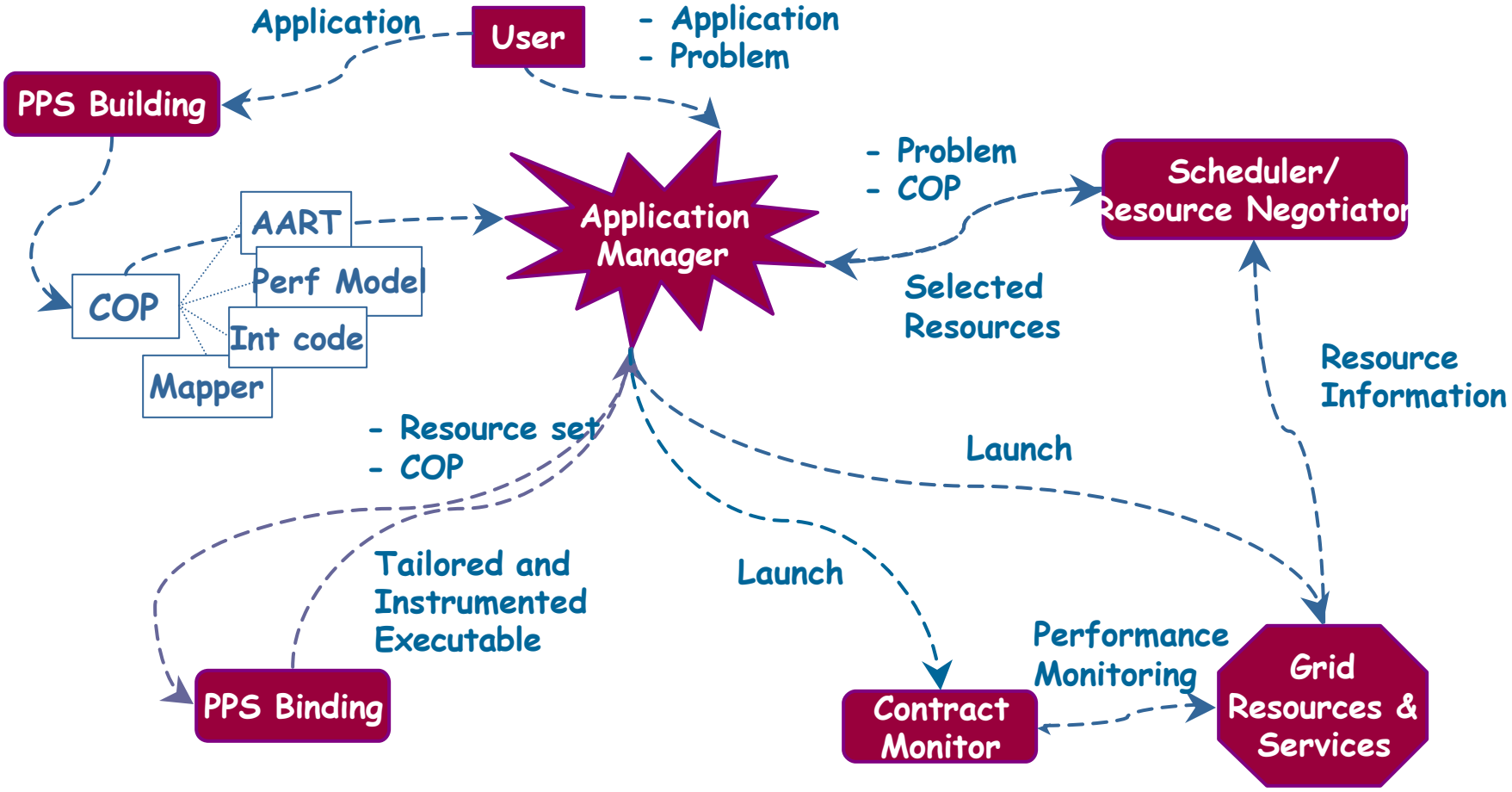
- GrADS project
  - Focus is on designing and developing a first prototype of a general, dynamic, usable, and scalable application execution system
  - Focus on performance of a single application

- Previous work provides an important context
  - Considerable research which assumes more or less about the target execution environment
    - AppLeS, NWS, NetSolve, Autopilot, etc.
    - Globus, PVM, Legion, I-Way, Ninf, etc.
    - Grid demonstration applications
    - Traditional scheduling literature, etc.

# GrADSOFT Execution Environment Research

- Preparation system/Execution environment integration
  - —discovery of application requirements and basic interaction between development and execution system
- Contracts
  - —development of formal specification method for performance requirements
- Scheduling
  - —development of basic integrated preparation system/execution system-aware automatic decision processes
- Monitoring & re-scheduling
  - —development adaptive control of application behavior and resource demands

# GrADSOFT Prototype



# Lessons Learned from GrADS

- Building Infrastructure is resource-intensive
  - Infrastructure investment for a smoothly working system is huge
  - Doing everything by hand is time-intensive and not scalable
- Complexity and Dynamism of Grid Environment are forces to be reckoned with
  - Complex to combine performance tolerances of each component to achieve performance of the whole system
  - Hand-offs between system components must not create excessive overheads
- Policies are required for a smoothly operating system
  - Performance of a single application may conflict with performance of other applications and/or resources
- Human infrastructure as important as software infrastructure
  - A tightly coupled research and development effort of the sort proposed in CGrADS is essential to the success of an effort of this size and complexity

# Close Interaction is Fundamental

- Design and development of new mechanisms for information and control flow between program preparation system, and the program and execution environment
  - —Information about the environment and program behavior in that environment must be discovered and communicated to program components in meaningful terms
  - —Program requirements must be communicated to execution environment in ways that admit to effective control

- Environment-aware program preparation and execution interaction is fundamental to achieve performance in scalable adaptive environments

# Information Calibration is Fundamental

- Information quality in Grid-environments must be factored into performance models, policies and contract negotiations
  - —Quantification of "quality" of resource and application performance of information needed to calibrate models and develop confidence level for predictions

- Authentic calibration of the "goodness" of parameters, models, information and predictions critical for achievement of performance in hard-to-predict environments

# Policy Required for Stability, Performance

- Design and understanding of the role of policy in adaptive, dynamic computational systems
  - —Competing resource and performance requirements of distinct applications and resources must combine to achieve acceptable performance for both systems and individual applications
  - —Policies must be developed, tested, and understood to ensure success of execution environment

- Experience with large-scale organizations demonstrates that well-thought-out and explicit policies are required for performance.