Embarrassingly Parallel Jobs Are Not Embarrassingly Easy to Schedule on the Grid

Enis Afgan
Purushotham Bangalore
University of Alabama at Birmingham

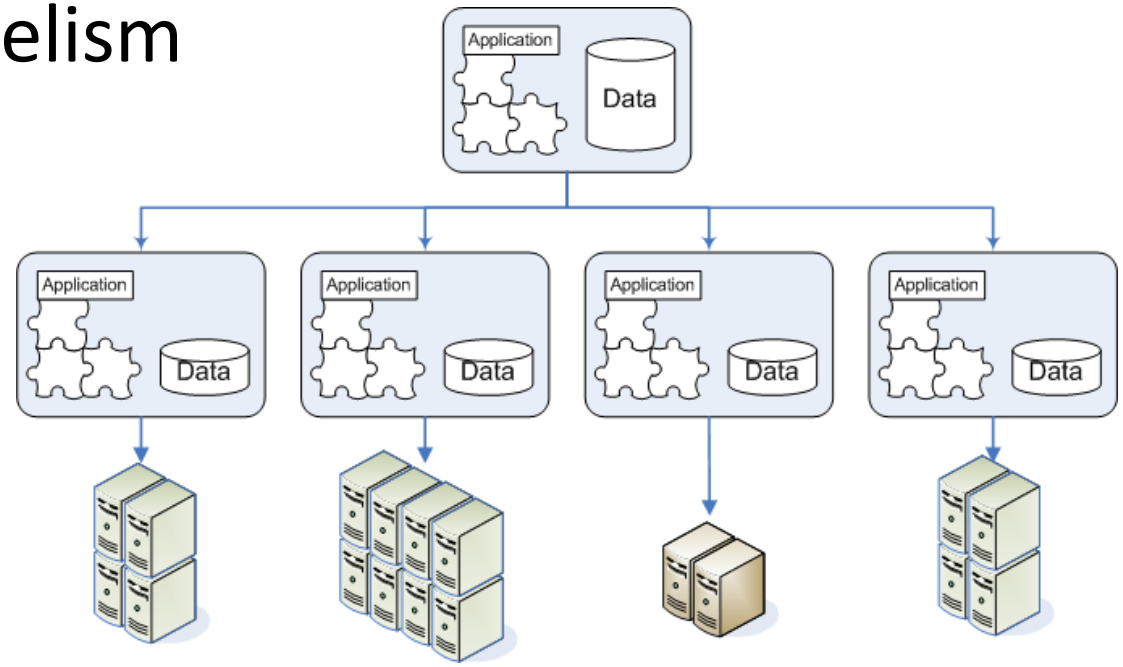
MTAGS 2008 Workshop at SC|08
Nov 17, 2008.
Austin, TX

Overview

- Embarrassingly Parallel (EP) Application Execution Model
- Execution and Scheduling Considerations
- Scheduling Examples
- EP Application Scheduling Framework
- Future Trends

Embarrassingly Parallel (EP) Applications

- Characterized by independent, coarsely grained, indivisible tasks
  - Similar to SPMD or Parameter Sweep Applications
- Introduce parallelism into application execution from a higher level

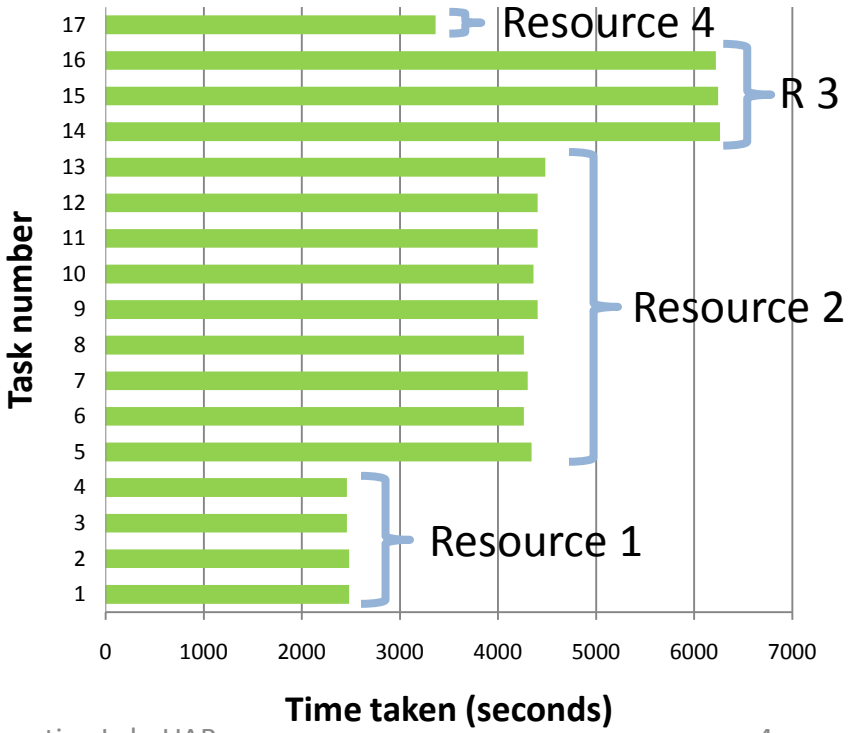


EP Application Execution Model

- A collection of tasks t_i working toward a common goal referred to as a job J:

$$J = \bigcup (t_1, t_2, \ldots, t_n)$$

- Execution in grid environments introduces task runtime heterogeneity
- Runtime of job determined by longest running task
- Goal: achieve load balance



Scheduling Considerations

- Load balancing
  - Static vs. dynamic, task granularity
- Job-level coordination
  - Development of a job plan, job plan flexibility
- Task failure
  - Job plan invalidation, redundancy techniques
- Cost
  - Resource consumer vs. resource owner
- User requirements
  - Accuracy vs. time vs. responsiveness vs. power consumption vs. availability

Task Parameterization

- Runtime of an individual task is influenced by a composition of several factors:

$$t_i = f\left(d_i, r_i, p_i\right) \begin{cases} d_i - \text{task input data} \\ r_i - \text{task execution resource} \\ p_i - \text{task invocation parameters} \end{cases}$$

- Achieving efficient execution of a job requires understanding and control over these components

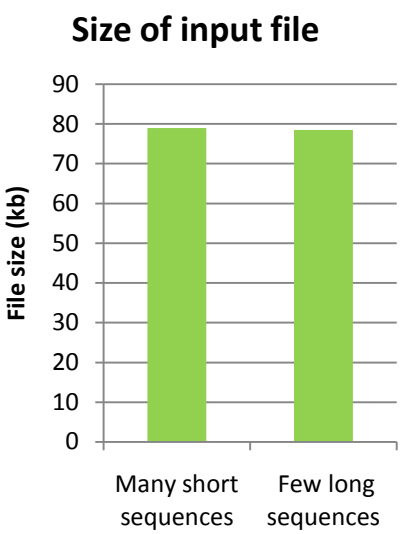
Task parameterization
(or understanding of components that are application, input data, and resource dependent)

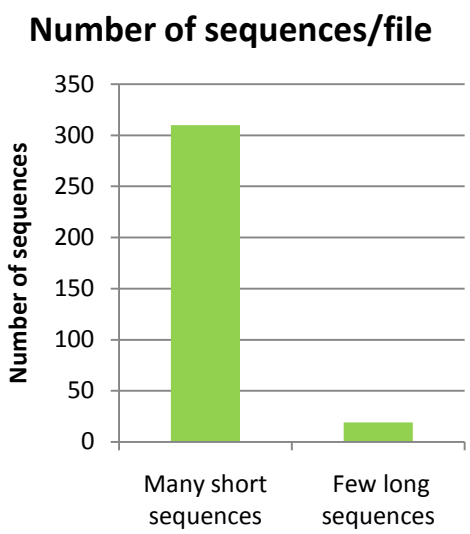
Leads to

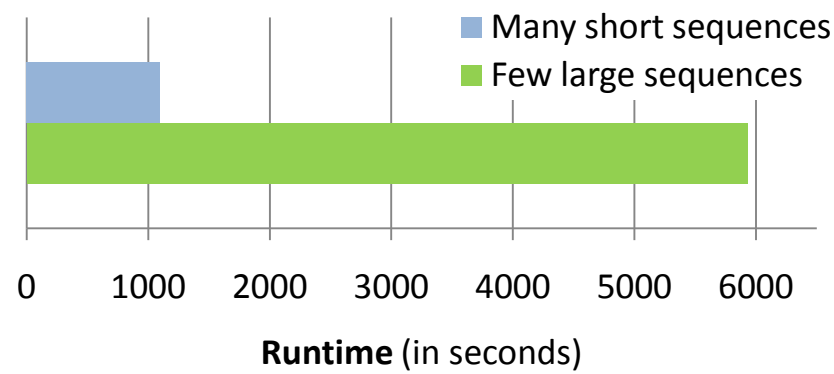
Fine level control over job execution characteristics

Understanding Task Parameters: Task Input Data (d_i)

- Primary mode of altering task workload
- Example: data (re)structuring
  - NCBI BLAST application performs biological sequence analysis
  - Size of input file is primary indicator of task runtime
  - Length of individual sequence has significant impact

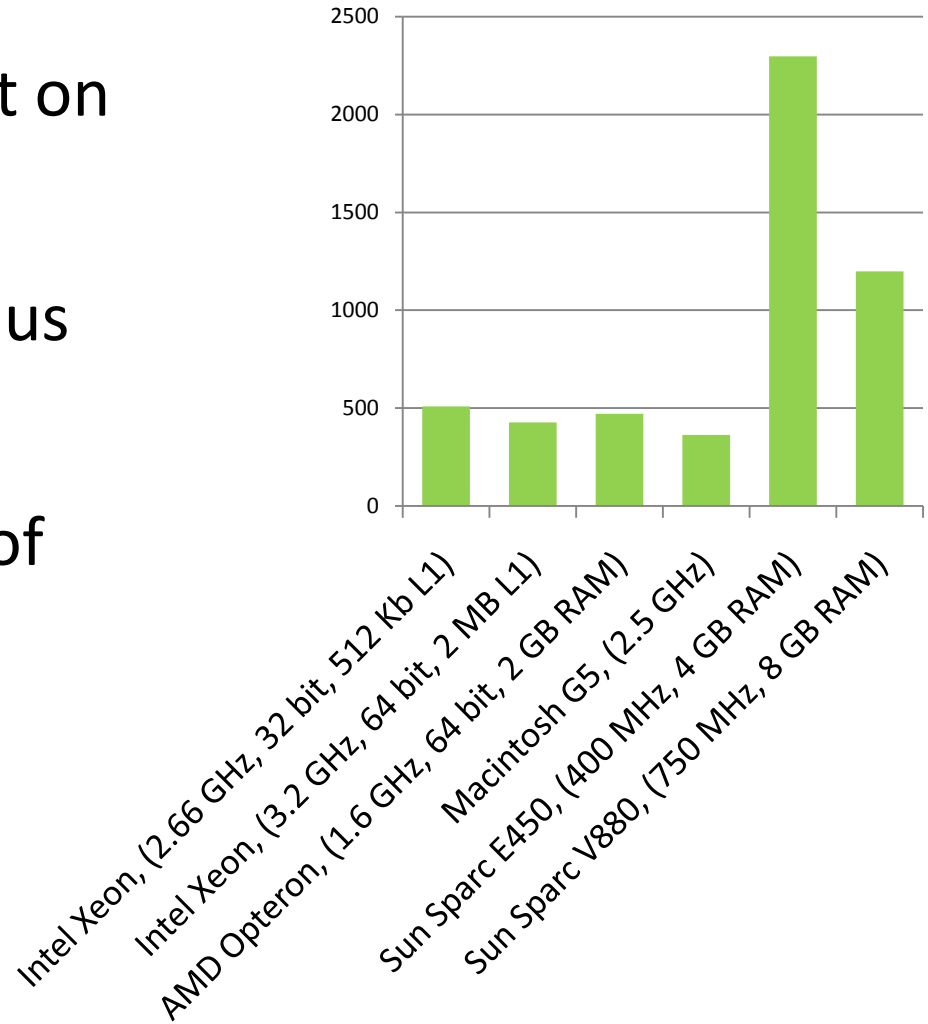






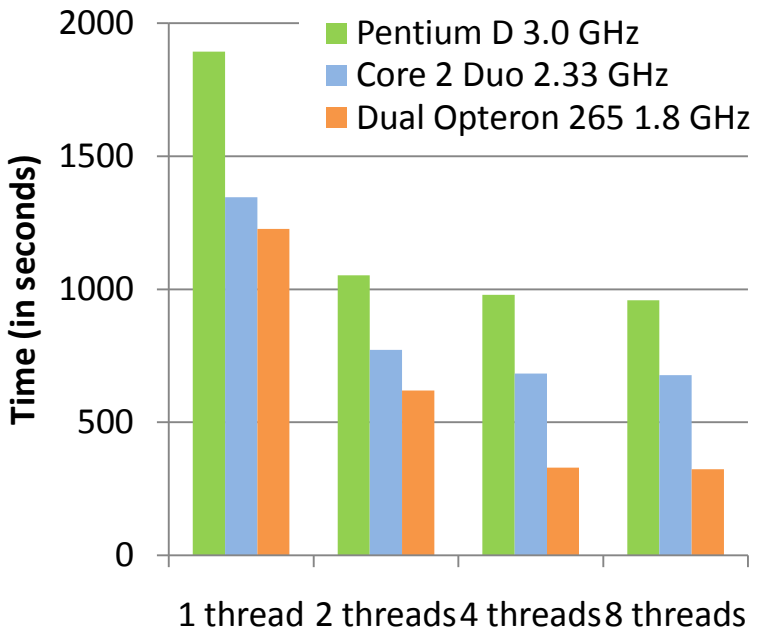
Understanding Task Parameters: Task Execution Resource (r_i)

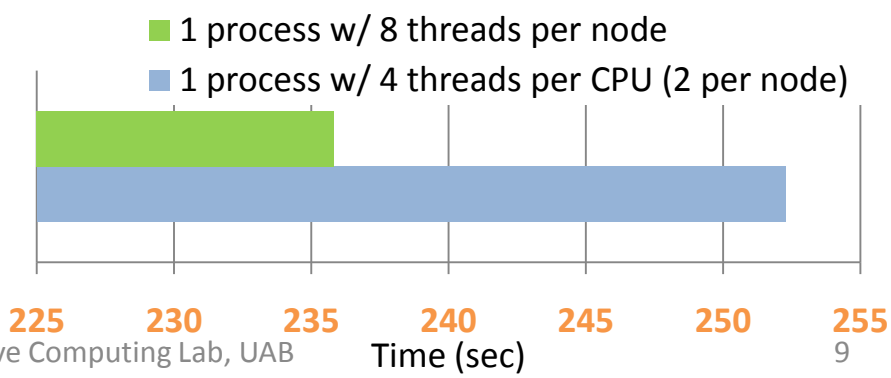
- Selection of execution resource has direct impact on task runtime
- Characteristic of the grid, resource heterogeneity thus offers most potential and flexibility
- The key is understanding of dependencies between application and resource
  - Hardware & software
- Example: BLAST across 8 architectures



Understanding Task Parameters: Task Invocation Parameters (p_i)

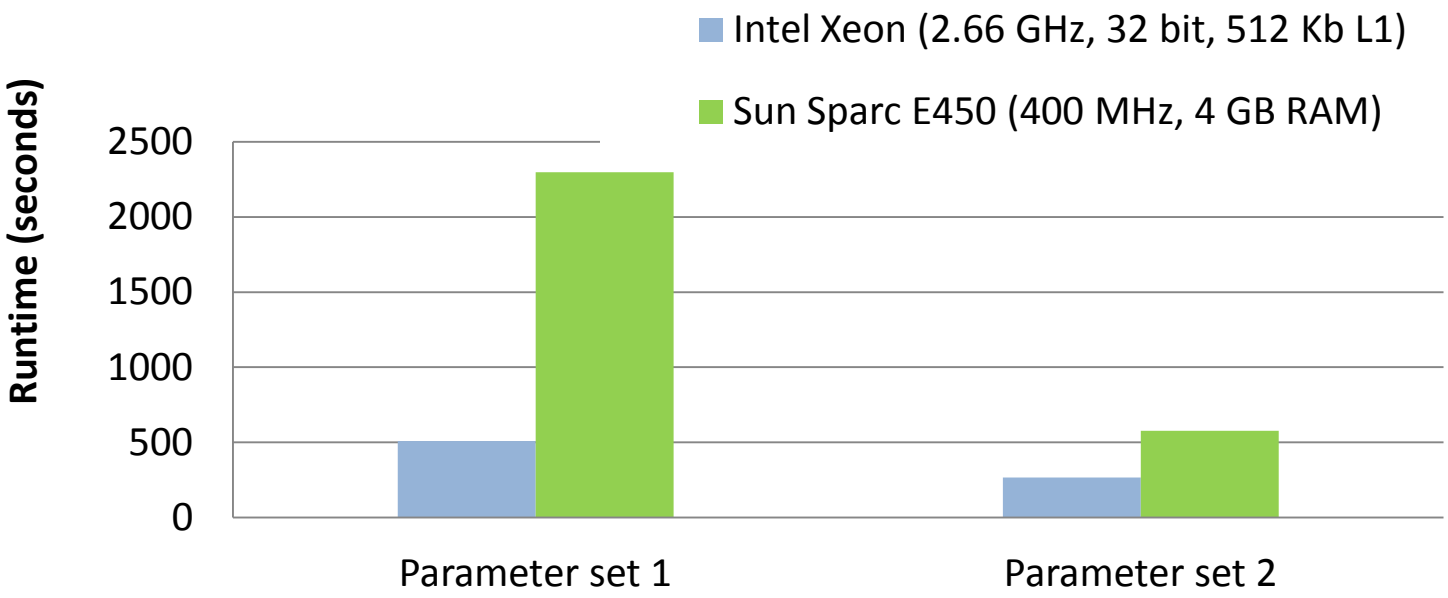
- Applications offer various invocation options and arguments that can affect execution characteristics (runtime, accuracy)
- Often application dependent:
  - Number of threads employed within a task, transitional probabilities in HMM, step size in MC simulations
- Knowledge about resource scheduling policies can be exploited to maximize task's efficiency: Start multiple processes/threads on a node





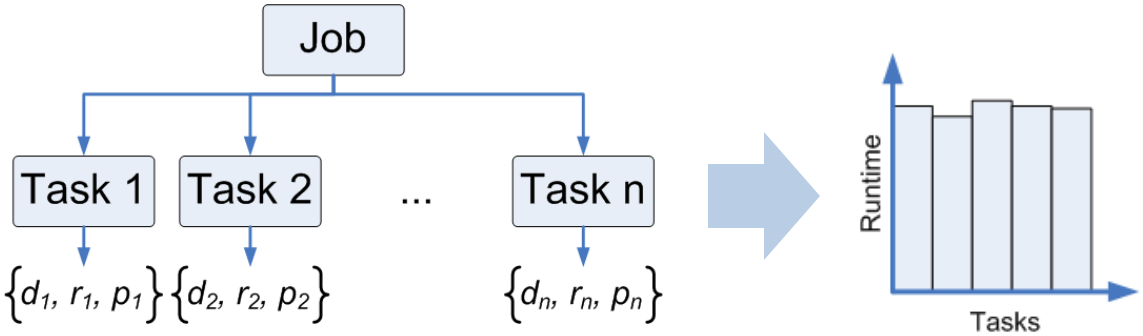
Influence of Task Parameterization

- Combining individual task parameters through task parameterization can result in fine level of control for task execution
- Sample impact:



Job Parameterization

- Above task parameterization, *job parameterization* operates at the task level



- Goal of most EP jobs is minimization of task load imbalance
- Job parameterization can introduce *parameterization dependencies* between tasks!
  - Regarding load balancing

EP Application Scheduling

- Execution of EP jobs in grid enables reduction in overall job runtime, increase in resource utilization and reduction in operating cost
- However, effective application execution is contingent upon recognizing, understanding and applying application-resource dependencies during job execution
  - Lack of understanding these dependencies can invalidate many benefits
- Understanding of this relationship can range from rudimentary to a complete mathematical model of application execution patterns
  - Drawback is that developing such an understanding can be a time-consuming effort with un-proportional effort for the benefit received

Scheduling Examples

- Two applications tested: NCBI BLAST and statistical R code
- For ALL the tests the same set of resources and input data was used
  - Increase in performance comes solely from understanding application-resource relationships and proper task and job parameterization
- Both applications were initially benchmarked and results studied
- Application-specific modules were developed to act on observed data

BLAST Performance Gains

Task definition: $t_i = f\left(d_i, r_i, p_i\right)$

d_i: Number of sequences assigned to a task

Initial:

$$d_i = |r_i| * \frac{D}{|r1| * |r2| * \ldots * |rn|}$$

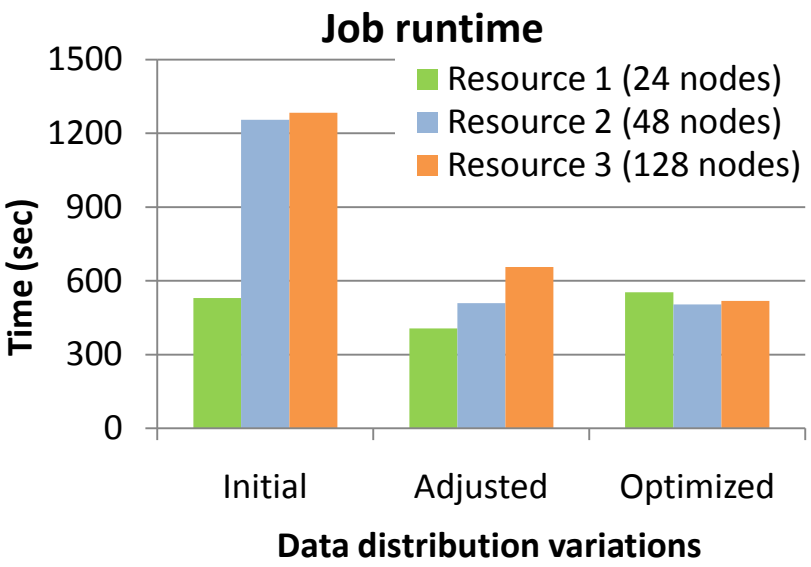
Adjusted & Optimized:

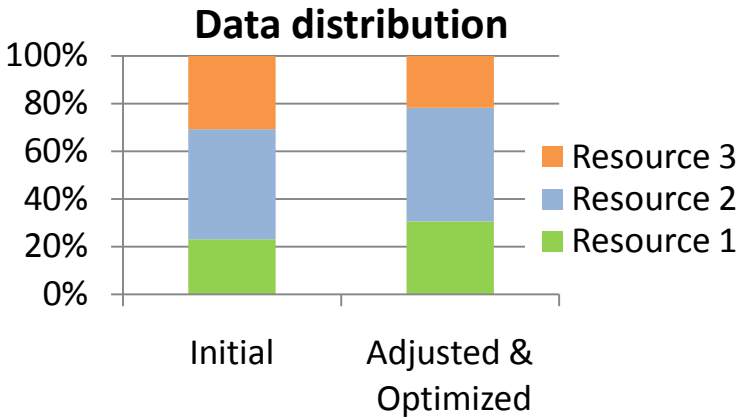
$$d_i = |r_i| * \frac{D}{|r1| * |r2| * \ldots * |rn|} * wi$$

Optimized variant further uses data reformatting at resource level.

r_i: Execution resource, the same for all cases

p_i: Exploits published resource scheduling policy by creating a single process and number of threads to match number of cores on given node.





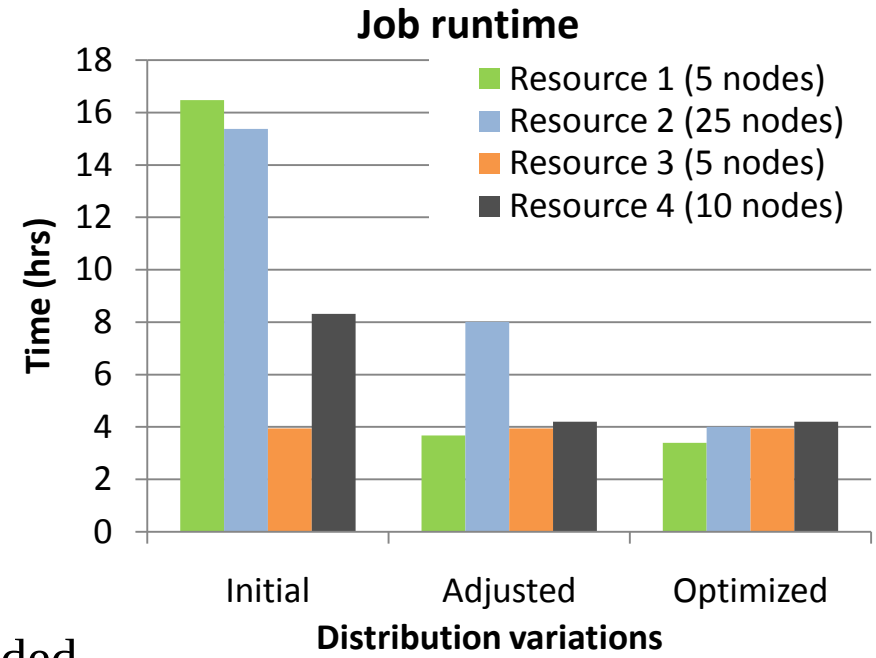
R Performance Gains

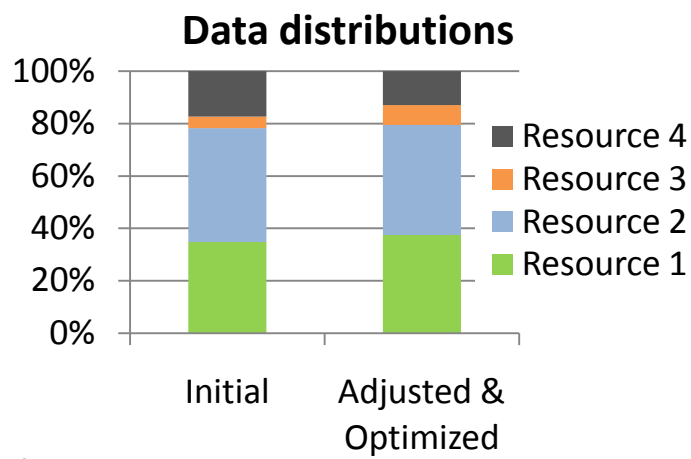
For data distribution, followed the same technique:

$$d_i = |r_i| * \frac{D}{|r1| * |r2| * \ldots * |rn|} * w_i$$

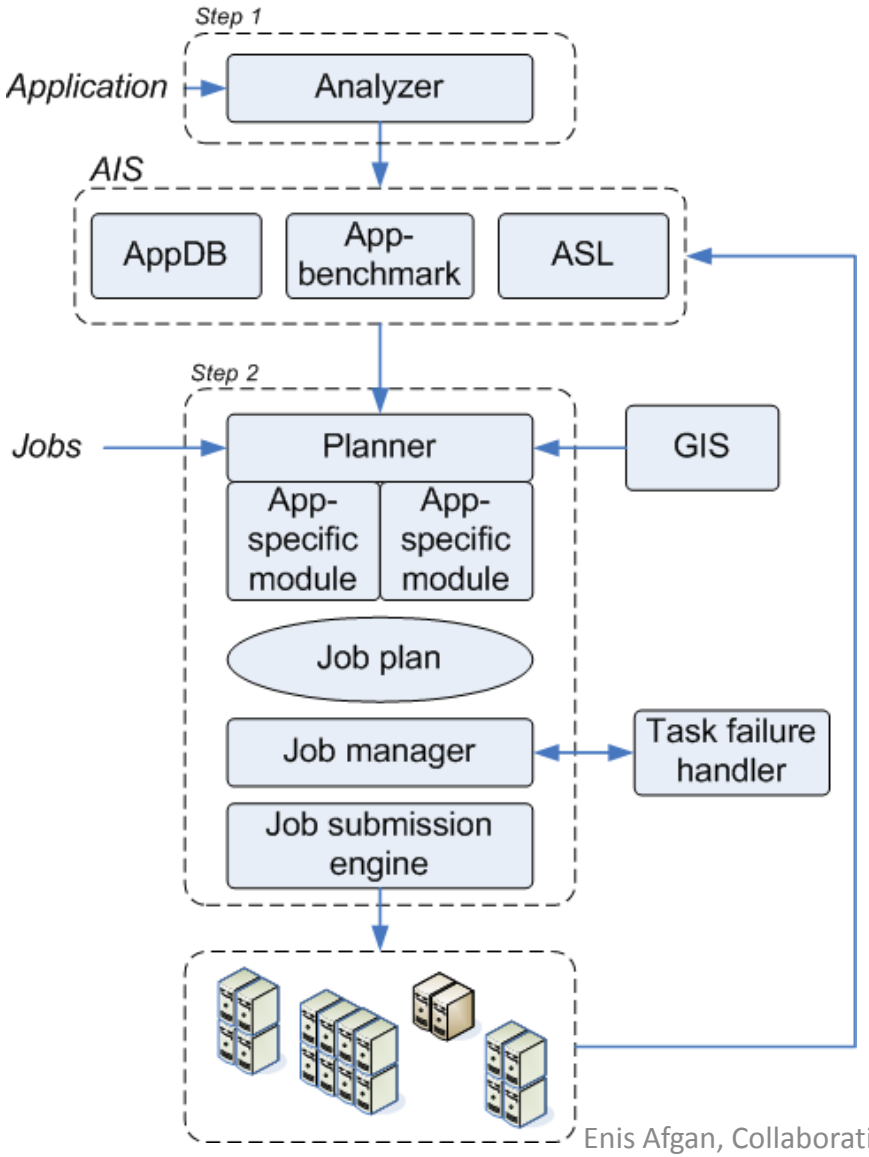
r_i: Execution resource, again, the same for all cases.

p_i: Unlike BLAST, R code is single threaded, so paramaterization focused on proper data distribution.





EP Application Scheduling Framework

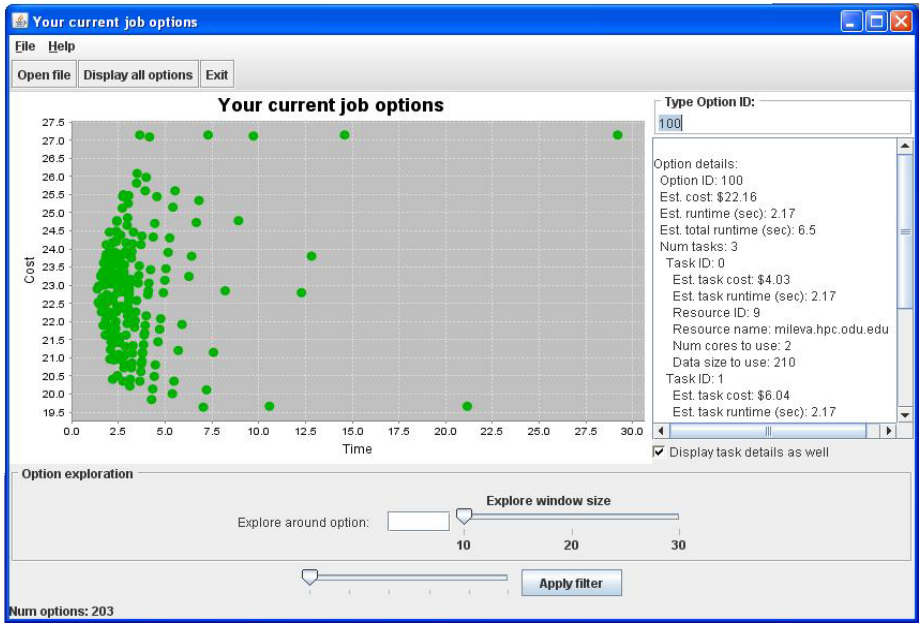


- Effective EP application scheduling is a two-step process:

1. Initially, the application needs to be analyzed, yielding needed application-specific information
2. Following, as jobs arrive, the derived information is exploited to account for the application-specifics and provide a job execution plan that aims at optimizing job's performance

Future Trends

- Full automation of the parameterization process
  - leading toward abstraction of job requirements from the user perspective
- Stepping away from job runtime minimization as primary system measure
  - Multi-objective scheduling considering cost, reliability, accuracy, responsiveness
- Provide a user with concrete job execution alternatives as a function of job parameters:



Summary

- EP applications represent a growing workload on grid resources
- Moving beyond initial execution of EP applications across grid resources, there is a need for efficient application execution
- Development of application-specific metascheduler modules is non-trivial task often requiring application and grid domain expertise
  - Provide hooks/plugins to existing applications to design “smart applications”
- Looking into the future – application of such schedulers to provide enterprise solutions targeting user- and job-specific customizations

Enis Afgan (afgane@uab.edu)
www.cis.uab.edu/afgane

www.cis.uab.edu/ccl

QUESTIONS?