Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments

Richard Huang, Henri Casanova, and Andrew A. Chien

Computer Science & Engineering and Center for Networked Systems, University of California – San Diego

Motivation

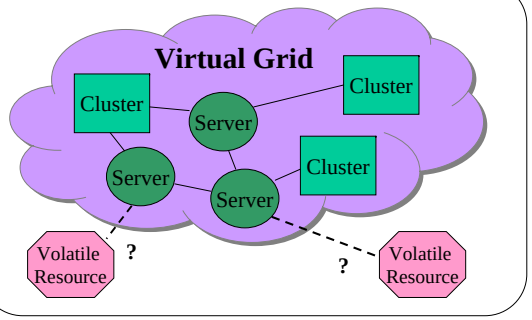
- There are lots of low-cost, volatile grid resources, which are attractive for applications; however, understanding application performance is challenging
- Applications face heterogeneity in resource
- Shared resources are dynamic in nature
- Application performance suffers further from variance in runtime prediction models
- Scheduling with different policies and parameters can impact performance by 2-5 times

Research Questions

- **Can we use volatile resources as part of virtual grids?**
- **How does dynamic resource load affect application performance?**
- **What scheduling policies should be used to account for or even exploit the uncertainties in application task runtime prediction?**
- **How does varying the task-to-resource ratio affect application performance?**

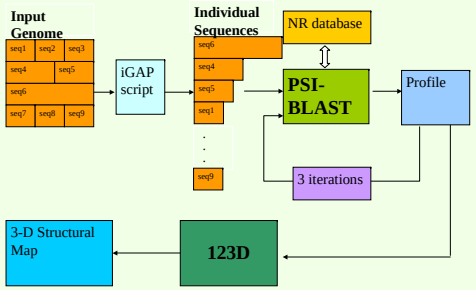
Goals

- Characterize volatile grid resources
- Understand how fresh dynamic resource data needs to be for better application performance
- Identify optimal scheduling techniques under different resource conditions
- Identify optimal task-to-resource ratios given scheduling policies



Experiments

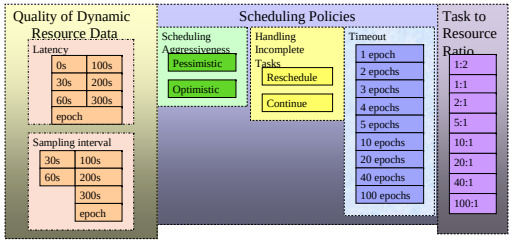
Applications (workload)



- PSI-BLAST and 123D represent a class of loosely-parallel applications typically found in production grids
- Run PSI-BLAST and 123D to generate actual task runtimes
- Create runtime prediction model by sampling task runtimes

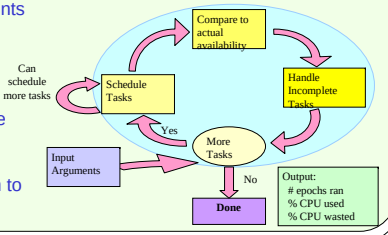
Approach

- Identify 12,096 scenarios to study, varying quality of dynamic resource data, scheduling policies, and task-to-resource ratio
- Acquire real task runtimes and resource availability data
- Simulate workload for each scenario using real task runtimes and resource availability data

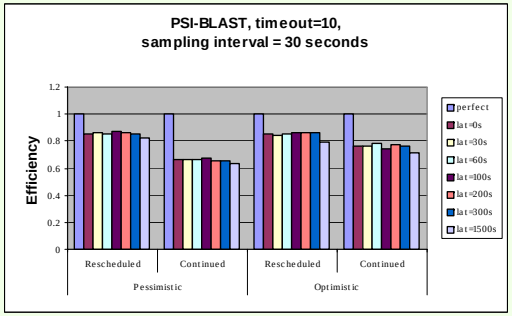


Simulator

- Simulation allows controllable and repeatable experiments
- Use actual task runtimes from PSI-BLAST and 123D
- Use real resource availability from trace data
- Use Max-min scheduling algorithm to schedule tasks

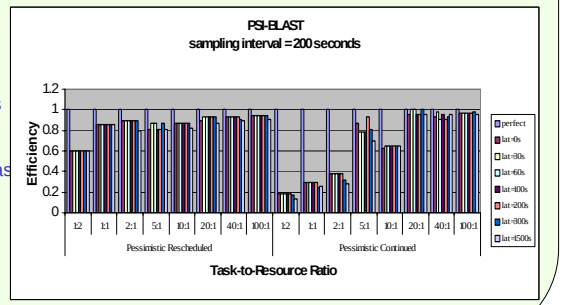


Results



- Freshness of dynamic resource data does not matter for any data fresher than 300 seconds
- Application performance varies less than 2% within this range
- Scheduling optimistically improves performance as the scheduler can tolerate volatility both in task runtime prediction as well as in the resource availability
- Performance can be further improved by rescheduling tasks running much longer than expected

- When resources equals or exceeds the number of tasks, handling of tasks that have greatly exceeded their predicted runtime is critical
- Rescheduling such tasks can improve performance by as much as 2 to 3 times
- When there are more tasks than resources, the opposite policy of allowing a late task to run to completion is preferred



Conclusions

- We can characterize some volatile resources (desktop grids)
- With appropriate scheduling policies (optimistic plus rescheduling), we can tolerate volatile resources
- We do not need the most up-to-date dynamic resource data for good performance
- Choosing rescheduling policies in volatile grid resource environments is critical

Future Directions

- Utilize volatile resources for virtual grids
- Make scheduling decisions based on monitored information for virtual grids
- Try hybrid approach of rescheduling after longer timeouts
- Perform cost-to-benefit analysis of using fresher dynamic resource data by determining costs for different quality of dynamic resource data
- Experiment with specialized scheduling techniques

Reference

Richard Huang. Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments. Master's thesis, University of California, San Diego, 2005.

Supported in part by the National Science Foundation under awards NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from the UCSD Center for Networked Systems, BigBangwidth, and Fujitsu is also gratefully acknowledged.