



Feedback-Controlled Resource Sharing for Predictable eScience

Sang-Min Park and Marty Humphrey

Dept. of Computer Science,
University of Virginia

This work was supported by the National Science Foundation under Grant No. SCI-0426972 and Grant No. SCI-0438263, the DOE Early Career program, and the Microsoft Research eScience Program.



Computer Science
at the UNIVERSITY of VIRGINIA



Emerging HPC Applications

- **Dynamic Data Driven Applications**
 - Linked Environments for Atmospheric Discovery (LEAD)
 - Earthquake modeling
 - Medical modeling for patient-specific treatment
 - Coastal hazard prediction, disaster planning, etc
- **Deadline-guaranteed Processing :**
Key to Success





HPC Infra. State-of-the-art

- Standard: Batch-mode, space-sharing
- Goal: predictable computation
- Research
 - Advance reservation
 - Queue wait-time prediction





Problem and (perhaps) Good Solution

- Space-sharing and related provisioning model cannot respond to adaptive computing demand predictably
- Good solutions must be able to -
 - 1) Change job's completion time at fine-granularity
 - 2) Cope with unanticipated disturbances





Our solution - overview

- Time-sharing principle
 - Focus is to control job's running time (not waiting time)
- Virtualization
 - Exploit VM's isolation / run-time reconfiguration
- Formal Control Theory
 - Provable correctness on deadline guarantee





Key Metrics and Assumptions

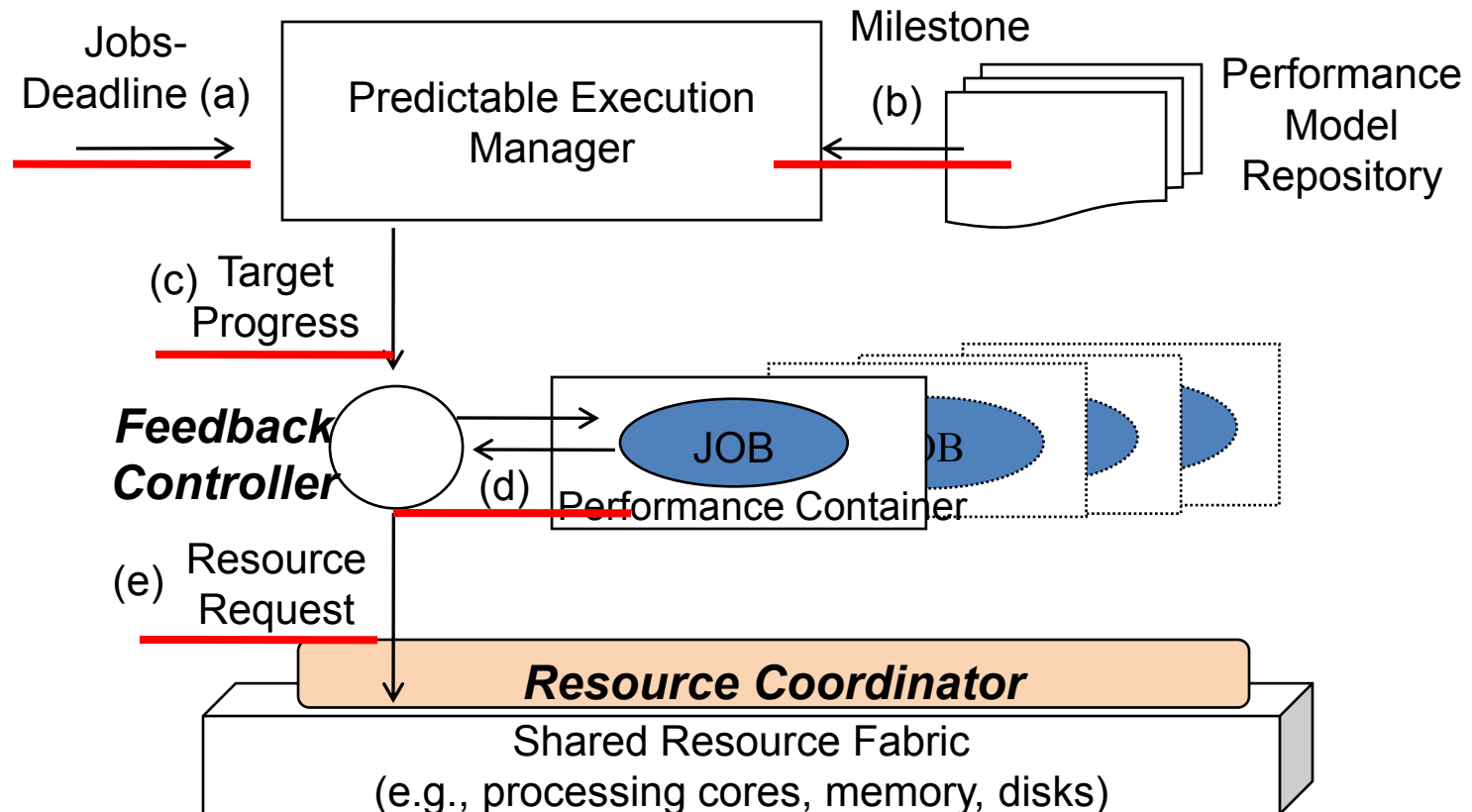
$$\frac{\textit{Milestone}}{\textit{Deadline}} = \textit{Progress}$$

- Milestone
 - Explicit overall computation
 - Examples
 - Total # of floating point operation
 - # of data file to process, # of query to process
- Progress
 - Computation in unit time (how fast)
 - Examples:
 - # floating point/sec, files/min., queries/min.





Compute Throttling Framework





Sensing - Actuation

- Sensing through source-code instrumentation
 - Sensor library via simple counter
 - Place sensor calls in critical path (e.g., after processing a file)
 - Alternative way for legacy support: profiling hardware performance counter
- Actuation through VM CPU scheduler
 - Both Hyper-V and Xen implements credit-based CPU scheduler
 - Use Hyper-V's scheduler cap to limit job's CPU consumption





System Identification

- Map resource consumption to progress
 - Question: how to know the right scheduling cap for keeping the progress?

- Use linear difference equation

$$S(k) = \sum_{i=1}^n a_i S(k-i) + \sum_{j=1}^m b_j C(k-j) \quad \longrightarrow \quad S(k) = aS(k-1) + bC(k-1)$$

- Black-box modeling approach

- Run jobs with varying CPU provision
- Use least-square regression to find model parameters





System Identification

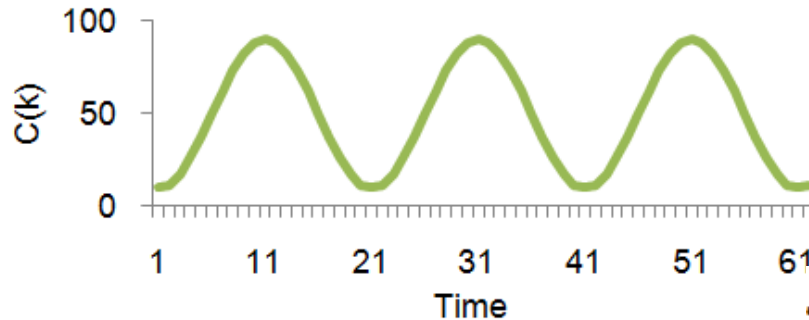


Figure 2(a): Actuation Values

BLAST

$$S(k) = 0.32 \cdot S(k-1) + 6.38 \cdot C(k-1)$$

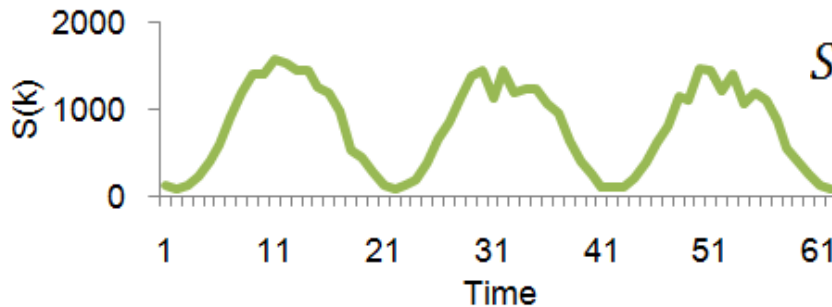


Figure 2(b): Sensor measurements from BLAST

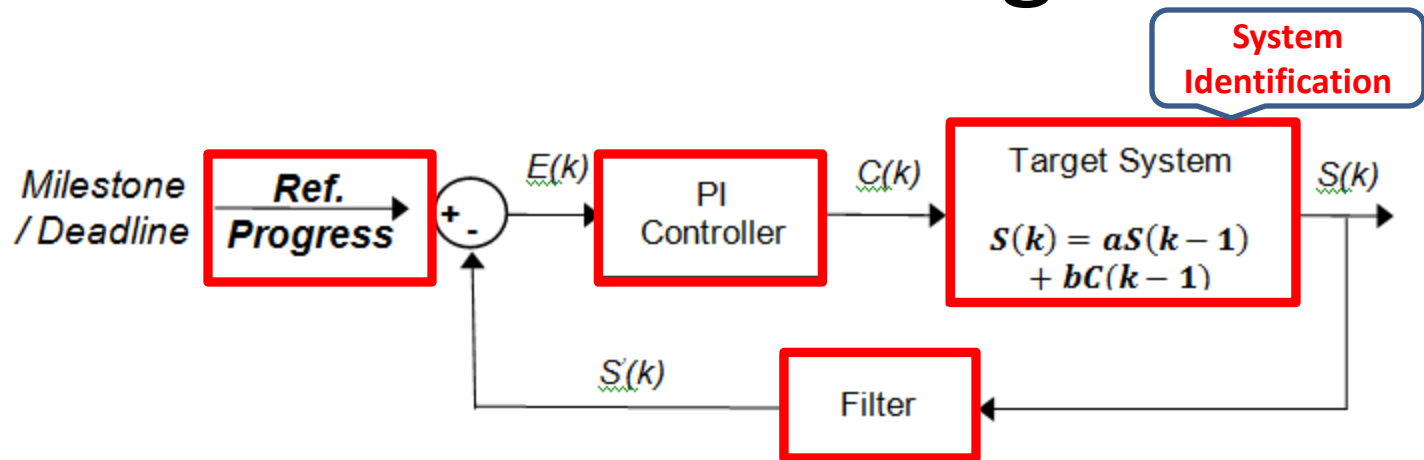
WRF

$$S(k) = 0.13 \cdot S(k-1) + 1.28 \cdot C(k-1)$$





Controller Design



- Closed-loop control copes with disturbances (e.g., model error, severe I/O)
- Control theory provides provable properties of closed-loop system
- PI control law in time-domain

$$C(k) = C(k-1) + (K_p + K_I)E(k) - K_pE(k-1)$$



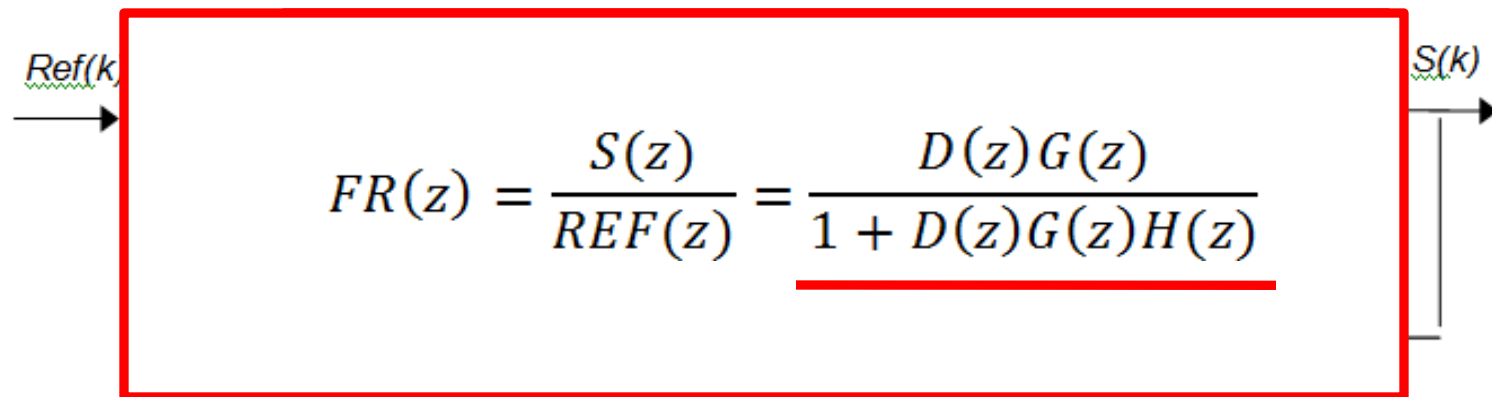


Controller Design – Cont'd

- Z-transformation: standard way of encoding and manipulating signals and system parts

$$D(z) = \frac{C(z)}{E(z)} = K_P + \frac{K_I z}{z - 1}$$

$$G(z) = \frac{S(z)}{C(z)} = \frac{b}{z - a}$$



$$H(z) = \frac{1 - c}{z - c}$$





Controller Design – Cont'd

- Find K_p and K_i in PI rule: $D(z) = \frac{C(z)}{E(z)} = K_P + \frac{K_I z}{z - 1}$
- Properties of PI-Control loop
 - Stability: must ensure closed-loop poles lie within unit circle
 - Accuracy: Integral-term guarantee zero steady-state error (100% accuracy)
 - Settling time $\approx \frac{\log \frac{k}{100}}{\log a}$, where a is the largest pole of the closed loop ($FR(z)$)
- Matlab provides control design tools for determining control parameters





Estimating Milestone

- How to estimate milestone
 - From application semantics (e.g., files to process)
 - From variables in source code (e.g., max iteration of loop)
 - Estimation wrt. quantifiable problem size

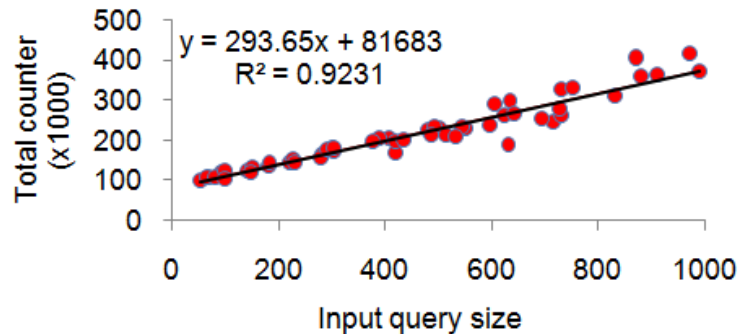


Figure 4 (a): Profiling results for BLAST

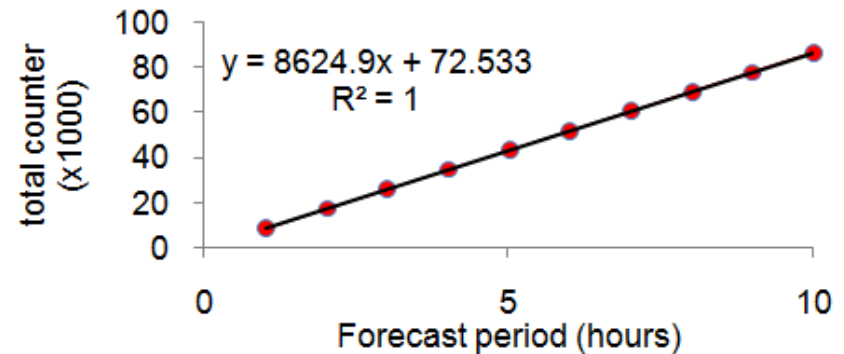


Figure 4 (b): Profiling results for WRF





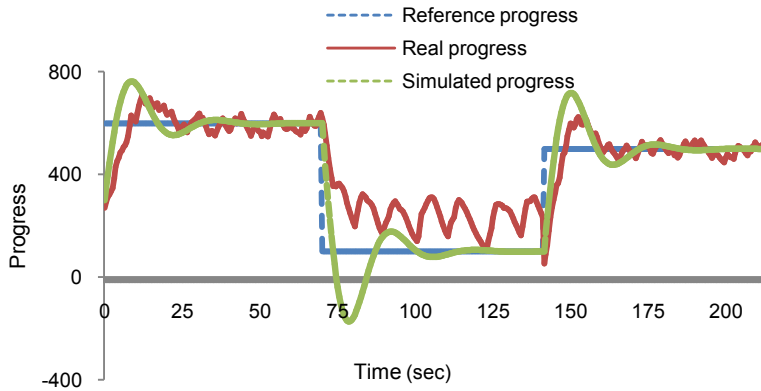
Evaluation - overview

- Applications
 - BLAST (sequence alignment) : data intensive
 - WRF (weather prediction) : compute intensive
- Questions to answer
 - Is feedback controller accurate?
 - Can it address run-time disturbances?
 - Does it meet deadlines?

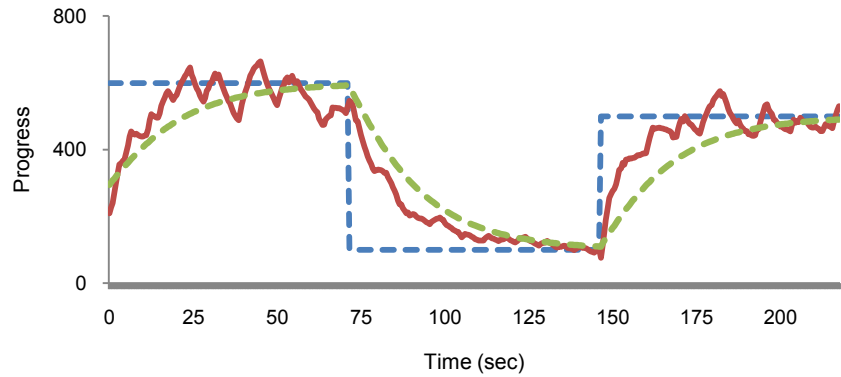




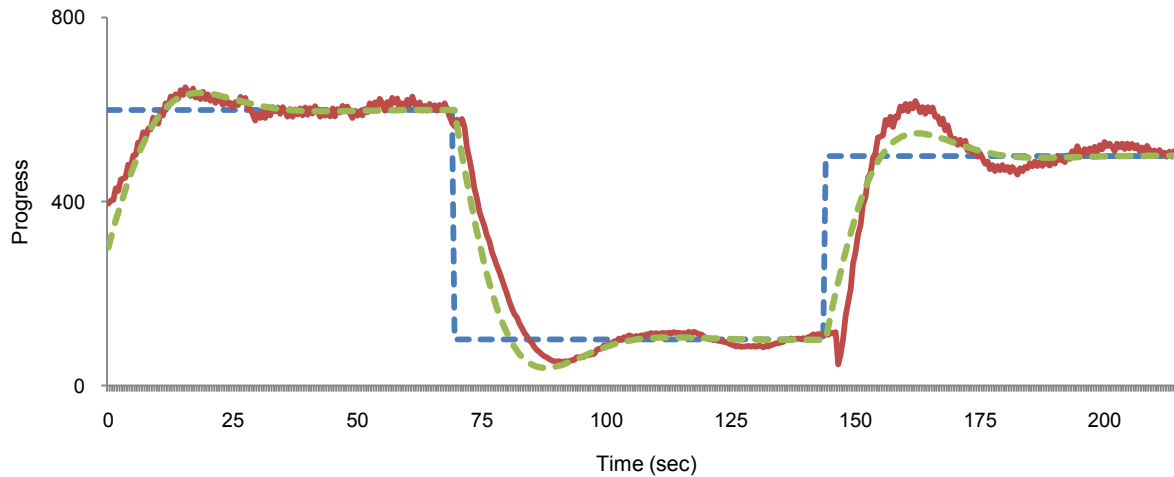
Evaluation – is controller accurate?



Aggressive (high-gain) controller



Slow (low-gain) controller

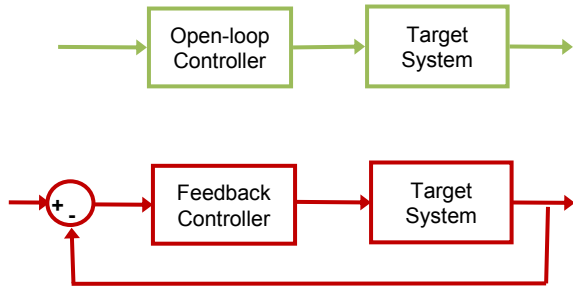


Good controller

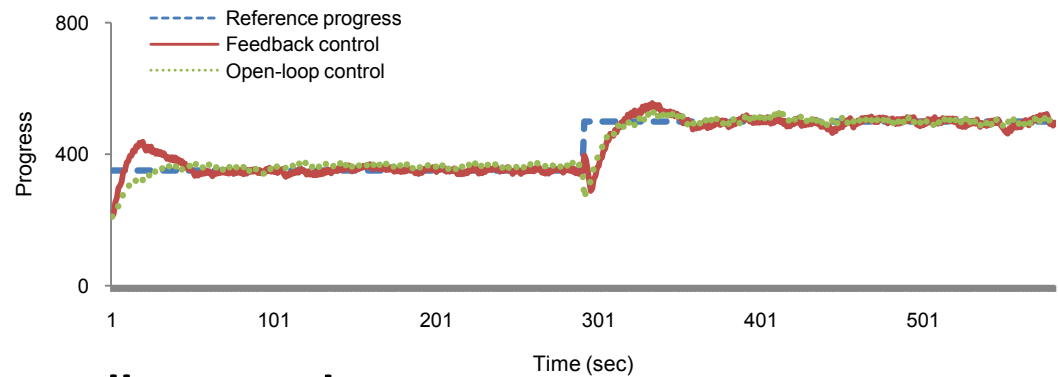




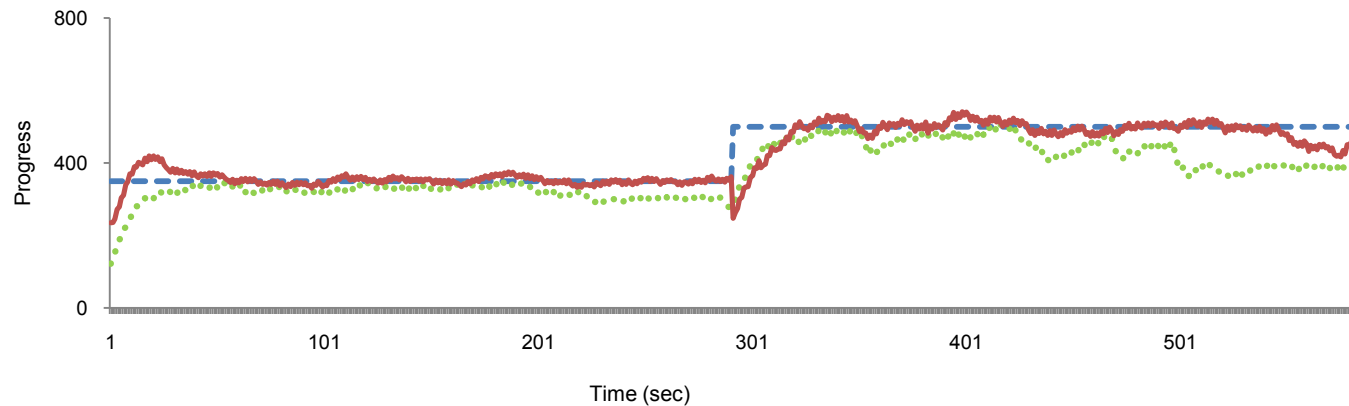
Evaluation – address disturbances?



No disturbance: feedback-loop = open-loop



Disk disturbance: feedback controller can adapt





Evaluation – deadline guarantee

- Goal: meet the deadline as close as possible
- Run **WRF** on AMD **8-core server**
- 8 best-effort jobs (VMs)
- 4 deadline-guaranteed jobs
- Two baseline
 - BATCH-RANDOM
 - BATCH-BEST
- Run for 8 hours (48 deadline jobs processed)





Evaluation – deadline guarantee

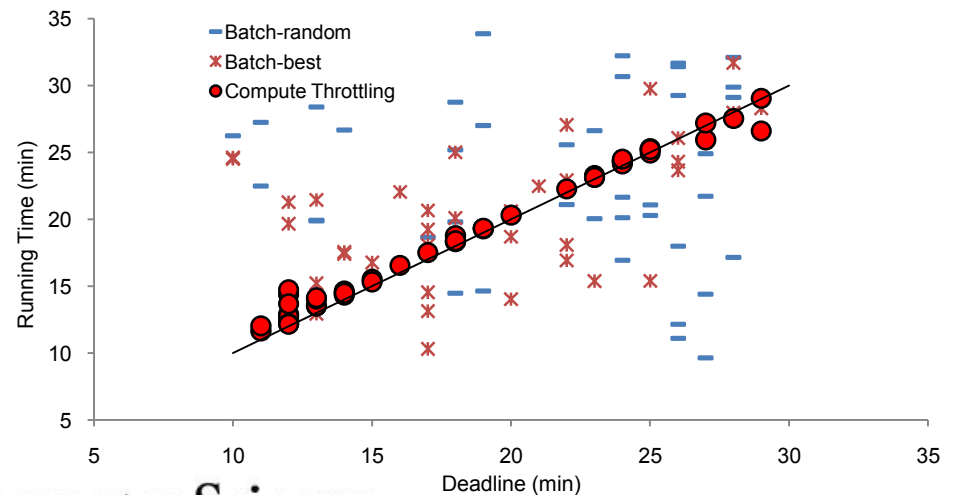
$$\text{Deadline-Guarantee Error} : \text{abs}\left(\frac{\text{running time} - \text{deadline}}{\text{deadline}}\right) \times 100$$

TABLE VI. COMPARISON OF THREE METHODS

	COMPUTE THROTTLING	BATCH BEST	BATCH RANDOM
Average Deadline-Guarantee Error	3.4 %	30.8 %	65.8 %
WRF Forecast Hours by Best-effort Jobs	1358 h	1470 h	1516 h

9 times higher accuracy than best case of batch-sharing

Overhead: About **90%** utilization of batch execution





Recent Efforts

- Question
 - Can we really expect for ordinary computational scientists to model applications and design controller?
- Self-Tuning Controller
 - System Identification, Controller Design, Actual Control all at run time
 - Usability is significantly improved
- More applications
 - ADCIRC (flood modeling), OpenLB (Lattice-boltzmann), Montage (astronomy workflow)



Recent Efforts

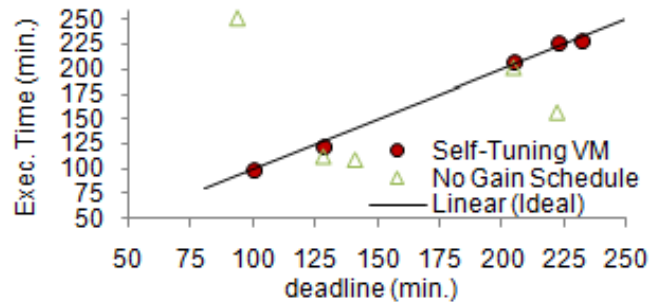


Figure 6(a) ADCIRC

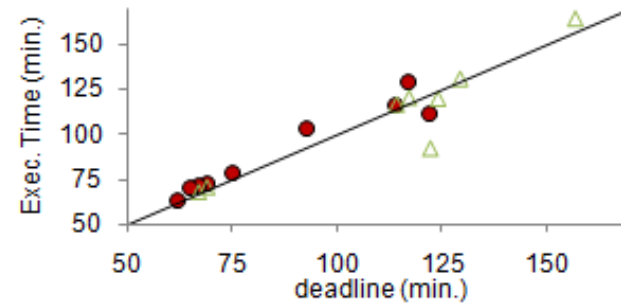


Figure 6(b) BLAST

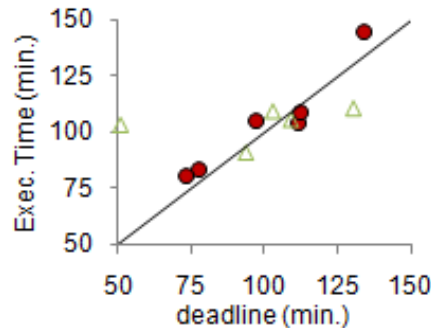


Figure 6(c) mProject

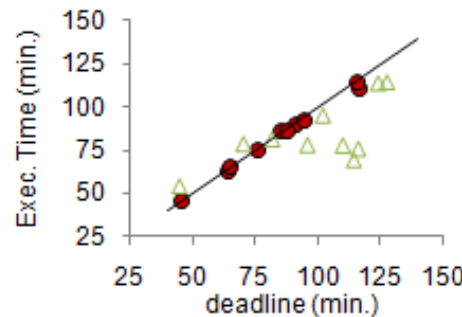


Figure 6(d) OpenLB

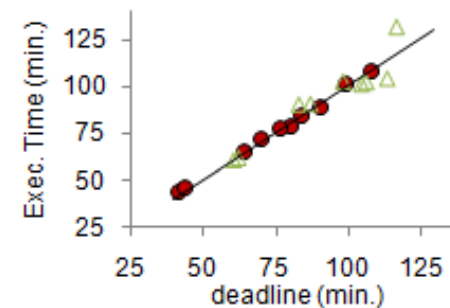


Figure 6(e) WRF

Programmed controller design in fact achieved better result than manual tuning!



Conclusions

- Batch-mode, space sharing cannot support time-dependent, adaptive computing demand
- We develop **controlled time-sharing** via VM
- **Control theoretic approach** achieves provable correctness
- Deadline can be met with less than 3% errors in general





Questions?

Sang-Min Park
sangmin@email.virginia.edu

