

Self-Tuning Virtual Machines for Predictable eScience

Sang-Min Park and Marty Humphrey

Dept. of Computer Science

University of Virginia

This research is supported in part by Microsoft Research



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 Statement

“Space-sharing and related provisioning model cannot respond to adaptive computing demand predictably”



Our solution - overview

Under
submission

SC'08, CCGRID'09

**Admission
Control**

**Time-
shared
Scheduling**

**Virtualized Resources
(CPU throttling, checkpoint)**



**Fully
Predictable
HPC
Infrastructure
(no batch
queue!)**



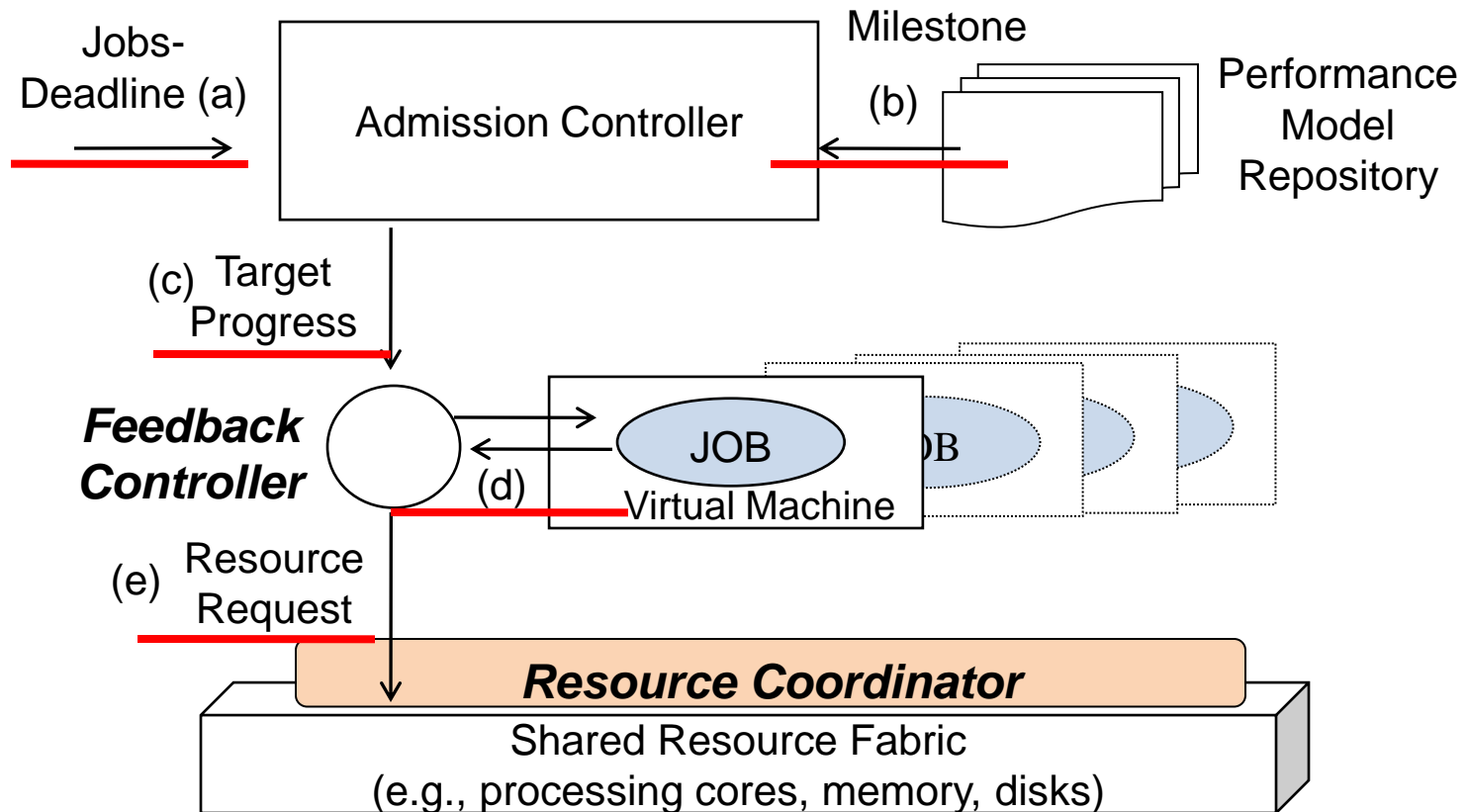
Performance Model: Metrics

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

- 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



HPC Server (VMM)



Why control theory?

- Well known in engineering (but not in CS)
- Feedback control copes with unknown disturbances (I/O, network, hardware upgrade, etc)
- Scheduler's properties are provable (via math)
 - Stability
 - Accuracy
 - Transient performances
- Active research in other areas
 - Real-time computing
 - Autonomic computing



Problems with control theory

- Who knows the theory?
 - Involves linear equations, z-transformations, modeling, Matlab, etc...
 - Require one or two semesters to learn
- Question that bothered us after SC'08:
“Can we anticipate an ordinary programmer to learn control theory to create a CPU scheduler?”
- Research question:
*“Can we **automate** the performance modeling and controller design process with no human support?”*

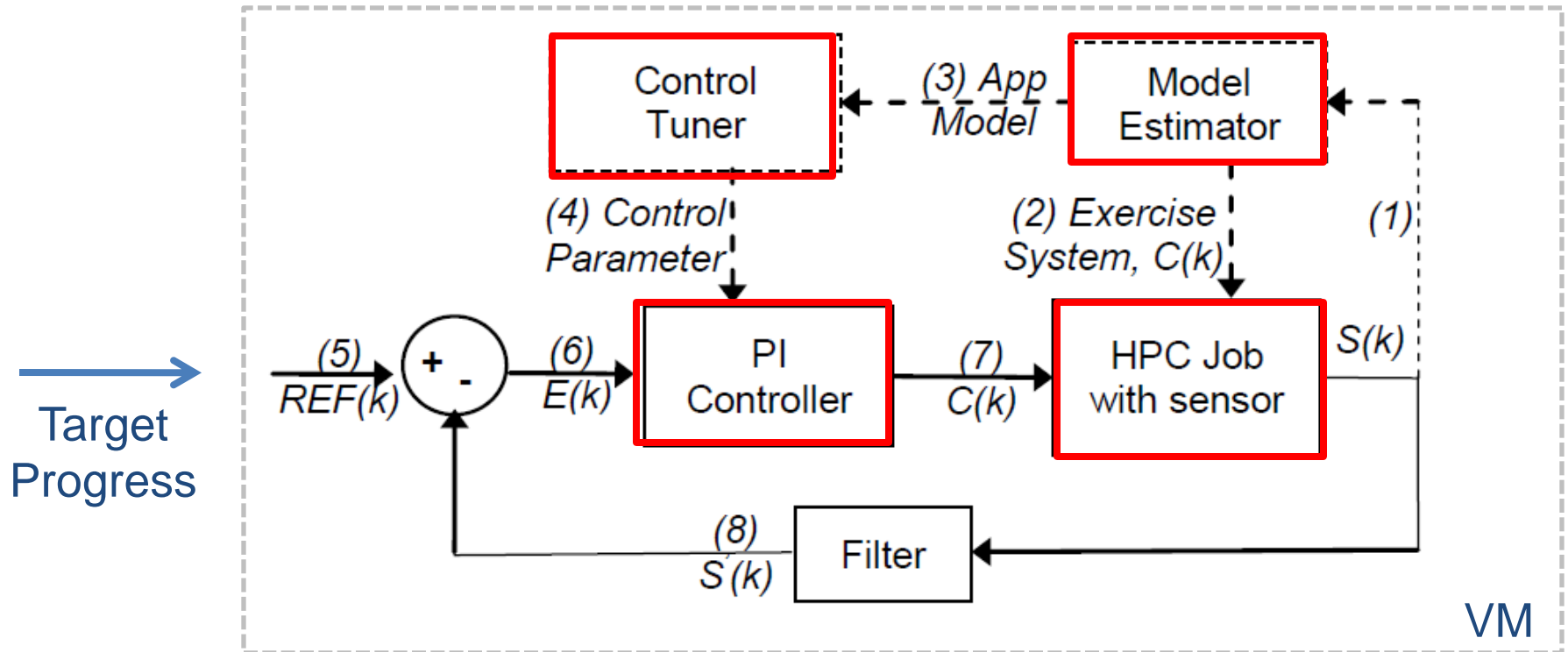


Self-tuning Virtual Machines

- Goal
 - User specify only the milestone (computation's quantity) and the deadline of the job
 - A HPC server (magically) completes the job before the deadline
- Approach
 - Perf. modeling and controller tuning is done at runtime, with no user support
 - So, customized schedulers are created runtime!

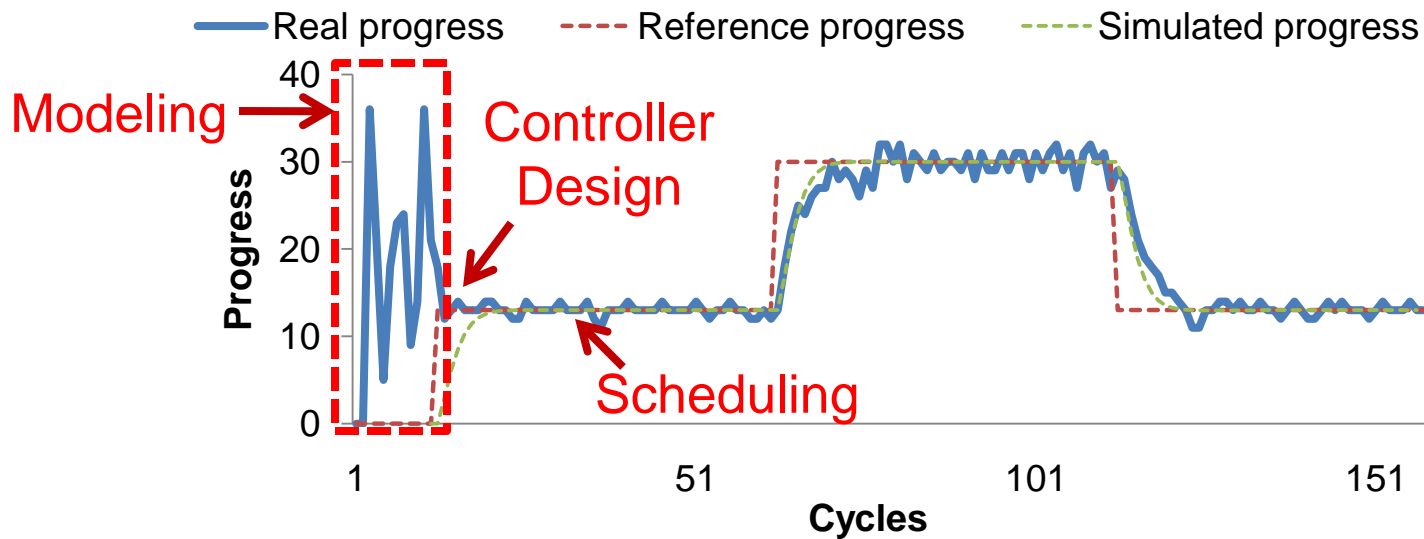


Self-tuning Controller



Self-tuning Controller

- Three phases in job's life-cycle
 - Performance **Modeling** by Model Estimator
 - PI Controller **Design** by Control Tuner
 - CPU **Scheduling** by PI Controller



Online Model Estimation

- HPC job with sensor
 - Source code instrumentation
 - Job reports progress every sampling period

- Performance model

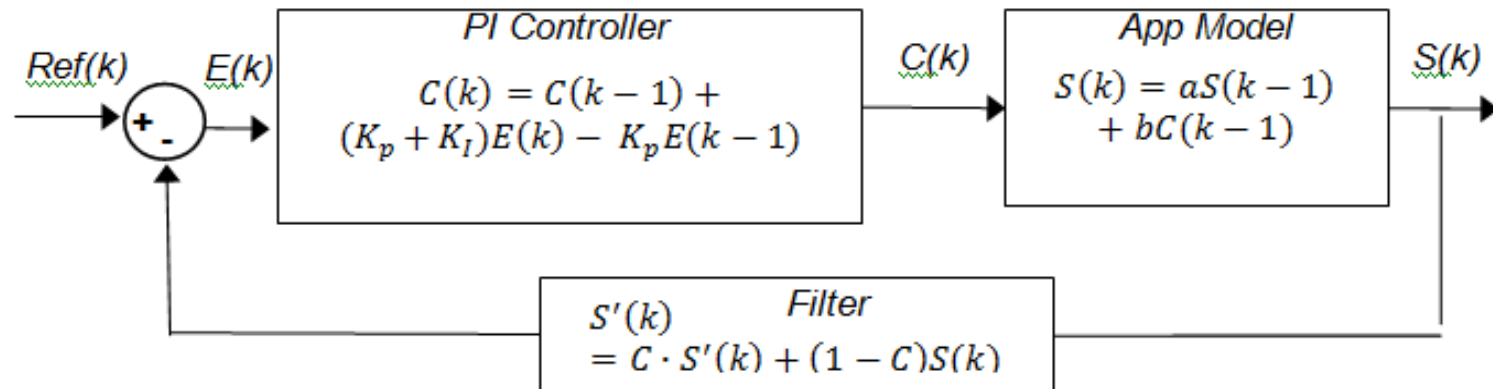
$$S(k) = \underline{a} \cdot S(k - 1) + \underline{b} \cdot C(k - 1)$$

- Model Estimator
 - Exercise the HPC job with varying VM CPU credit
 - Least-square regression on $(S(k), C(k))$



Feedback Loop with PI Controller

- Goal: keep the job's progress at the target
- Controller's K_p and K_I determines the behavior of closed-loop

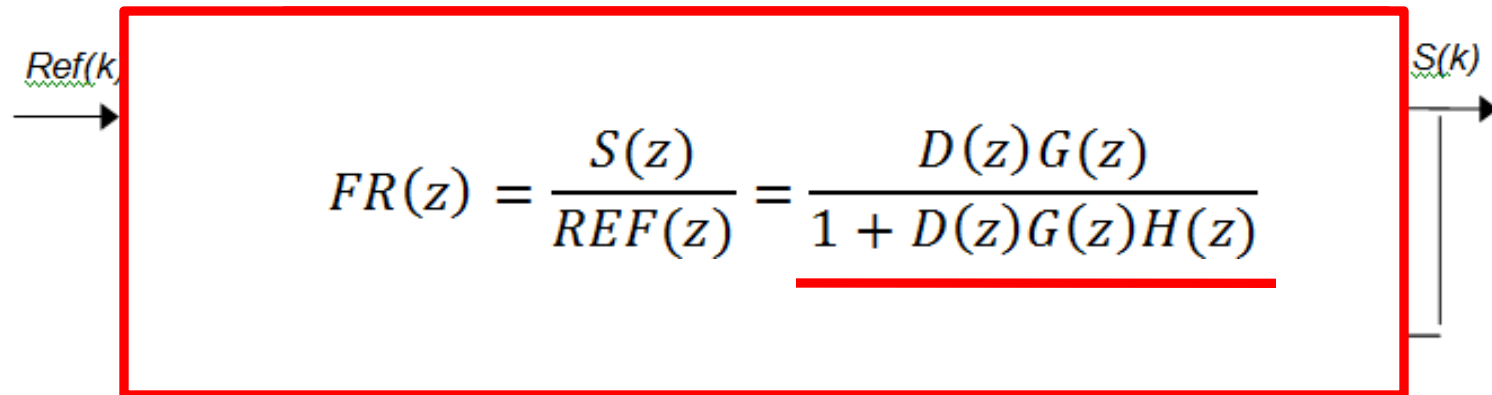


Feedback Loop with PI Controller

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

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

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



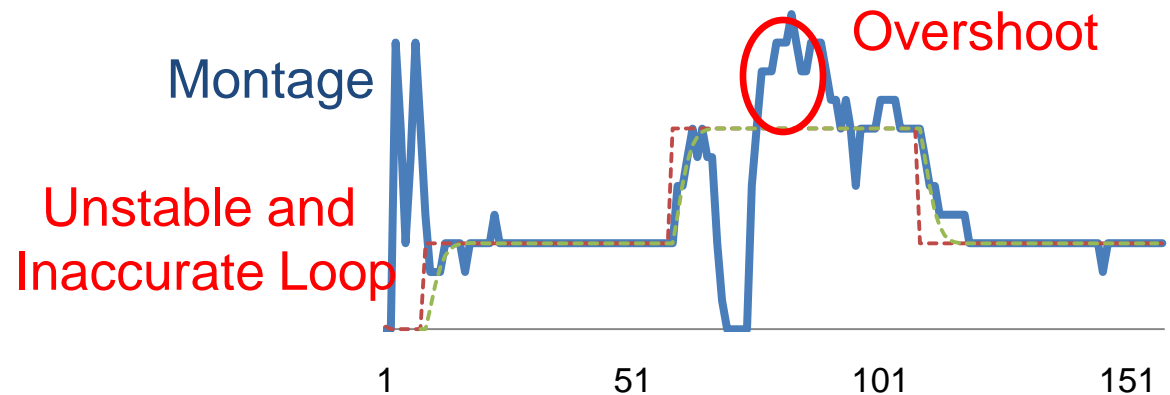
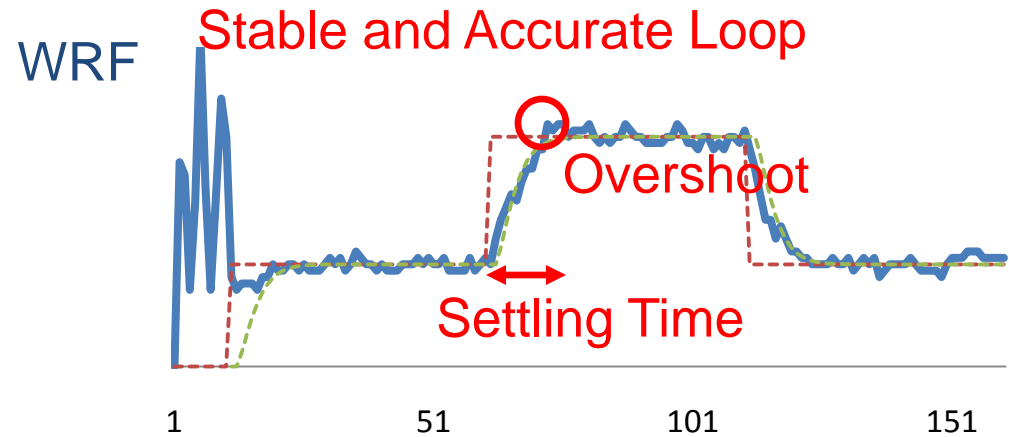
$$H(z) = \frac{1 - c}{z - c}$$



PI Controller - properties

- SASO Property

- Stability
- Accuracy
- Settling Time
- Maximum **O**vershoot



PI Controller - properties

- Control theory predicts loop's properties
- Stability
 - Closed loop is stable *iff* all poles of closed-loop are inside the unit circle (≤ 1.0)
- Accuracy
 - Zero steady state error due to Integral term
- Settling Time
 - $K_S \approx \frac{\log \frac{k}{100}}{\log a}$, where a is the largest pole of the loop
- Maximum Overshoot
 - $M_P \approx r^{|\theta|}$, where $r = \sqrt{c^2 + d^2}$ and $\theta = \tan^{-1}(\frac{d}{c})$



Auto-tuning PI Controller

- Online algorithm to satisfy SASO properties
- Infinite possible values of K_p and K_i
- Basic strategy: Parameter search
 - Choose arbitrary K_p and K_i and check SASO properties using the theory
 - But Z-transformation is computationally expensive!
 - Trade off between quality and running time of search algorithm



Auto-tuning : Heuristics

- Two heuristics to prune the search space early
 - Use minimum pole of the closed loop
 - Use stability analysis to prune the range
- Heuristics result in fast running time with near-optimal controller quality
- Rank algorithm
 - Find the K_p and K_i that results in minimum settling time and moderate overshoot



Evaluation – Usability

- Steps to create a predictable HPC system
 - (1) Instrument source code with sensor library
 - (2) Estimate milestone of a job instance
 - (3) Create a performance model of the job
 - (4) Design a PI controller for the new model
 - (5) Run the job with the new controller (CPU scheduler)
- Experiences from 5 HPC applications
 - Montage: very small effort
 - ADCIRC, OpenLB: small effort (few hours)
 - BLAST, WRF: moderate effort (a day or few days)

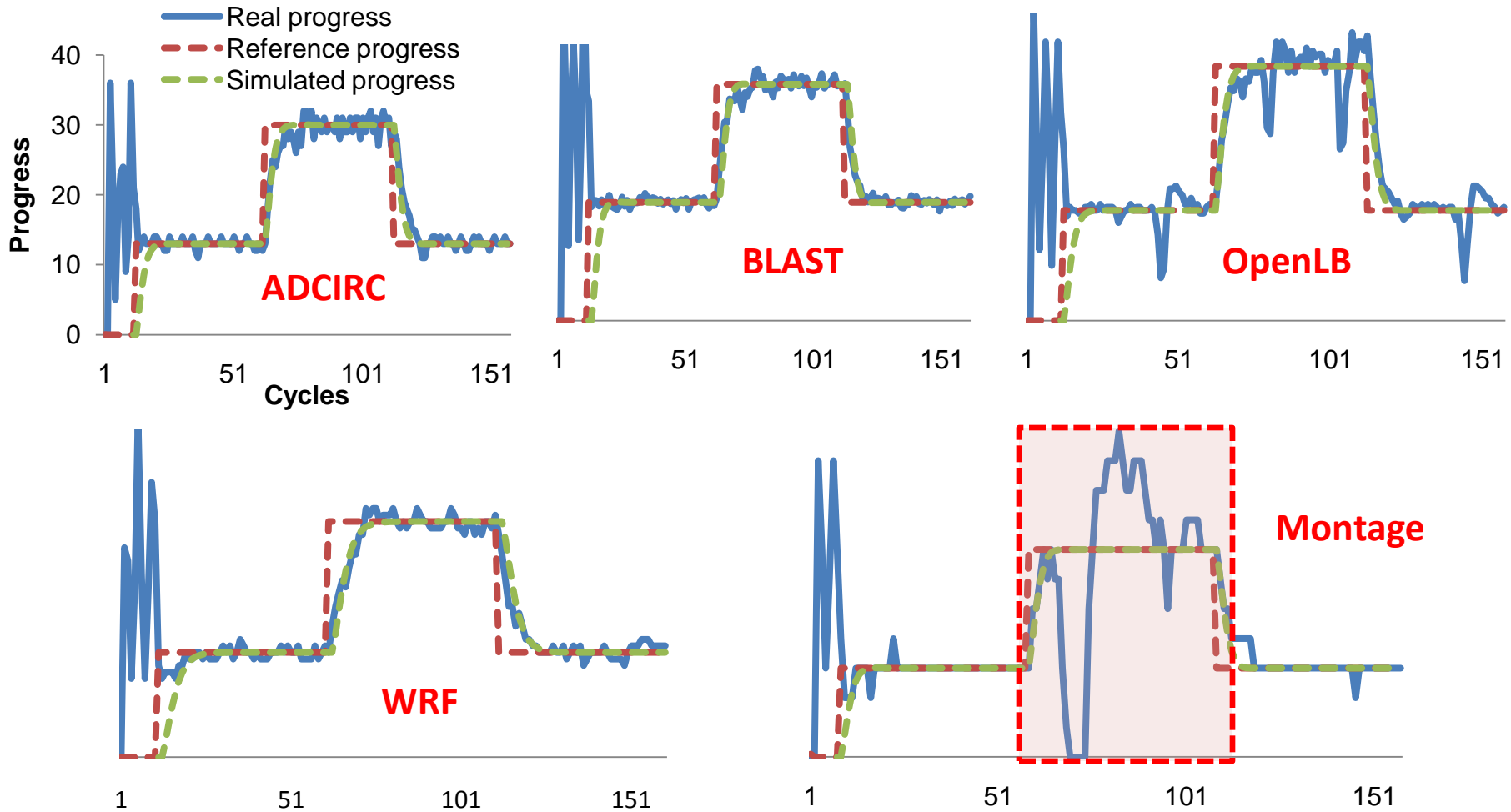


Evaluation - Performances

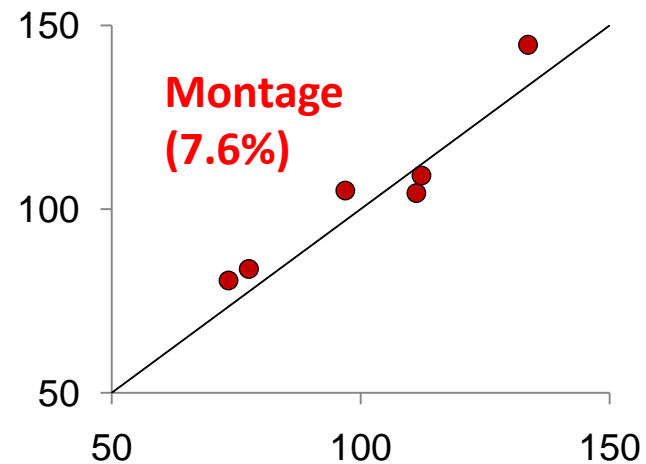
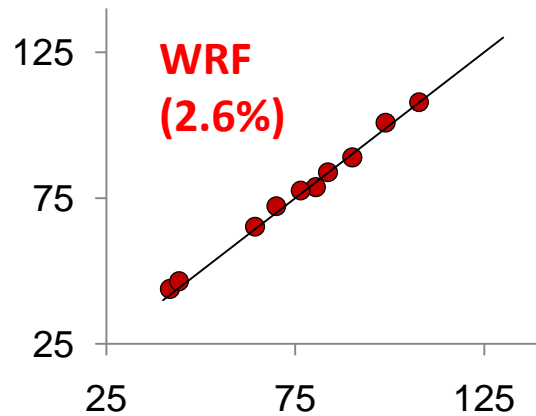
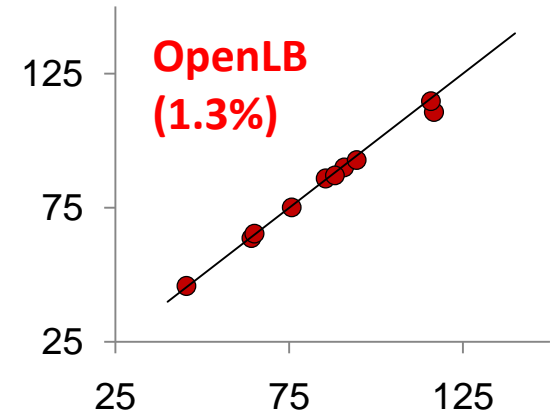
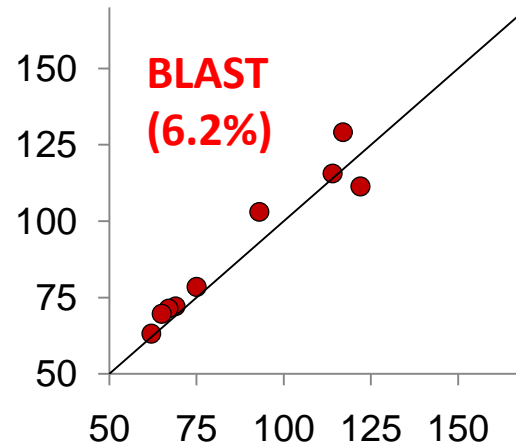
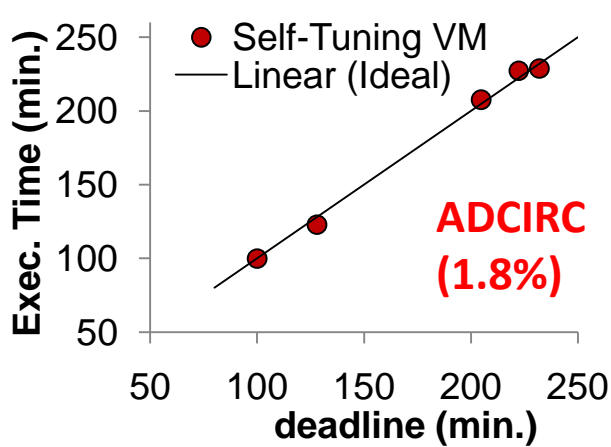
- Five Applications
 - **ADCIRC, OpenLB**: Fluid-flow modeling, compute intensive
 - **WRF**: Weather forecast, compute intensive
 - **Montage(mProject)**: Image mosaic engine, data Intensive
 - **BLAST**: Biology, data Intensive
- HPC Server
 - 8-core AMD server
 - Microsoft Hyper-V as VMM



Correctness of online tuning



Meeting deadlines (multiple jobs)

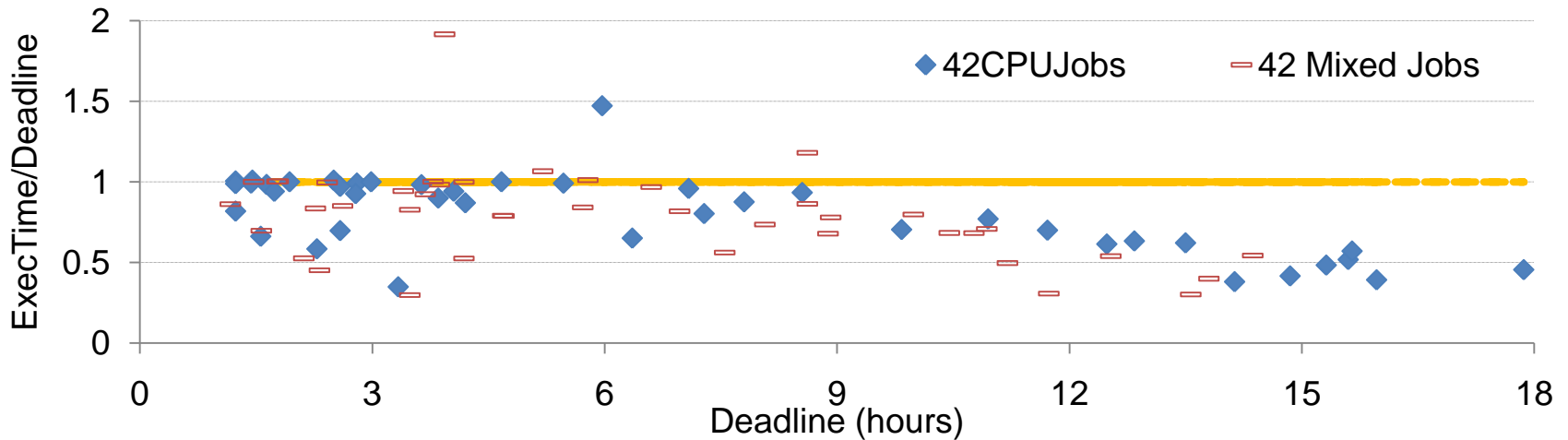


Recent works – Admission Controller

- Assumptions in this paper
 - User's requested deadline is not too early (can be met with 100% CPU consumption)
 - # of deadline-guaranteed jobs < # of CPUs
- Admission controller
 - Makes accept/reject decision regarding deadline, concretely at submission time
 - No restriction on deadline, # of jobs
 - Rely on VM's checkpoint for memory scheduling, in addition to feedback-control CPU scheduler



Admission Controller – Early Results



- Over 90% jobs met deadlines
- Only 12% errors for late jobs
- IO-intensive jobs cause some problem: still over 80% of success
- **Queuing system may be replaced by admission controller!**



Conclusion

- Control theory offers provable stability, accuracy on scheduling time-sensitive jobs
- Yet, the theory was not easily accessible
- Self-tuning VM fully automates modeling, controller tuning, and scheduling
- Admission controller and parallel jobs are the next topic of research



Questions?

Sang-Min Park
sangmin@email.virginia.edu

