

CRAM: A Congestion-Aware Resource and Allocation Manager for ...

Randal Burns

John Linwood Griffin

Johns Hopkins University



HEC FSIO R&D Conference, 11 August 2009

CRAM: Project Overview

....for Data-Intensive High-Performance Computing

- Problem: Processor-centric resource management wastes huge amounts of HPC resources
 - Congestible (shared) resources often limit performance
 - Leads to poor utilization of expensive facilities
- Goal: Allocate and schedule all of a system's heterogeneous resources
 - Detect and avoid congestion
 - Balance resource usage to maximize system throughput
 - Exploit workload elasticity to reconfigure jobs spatially and temporally



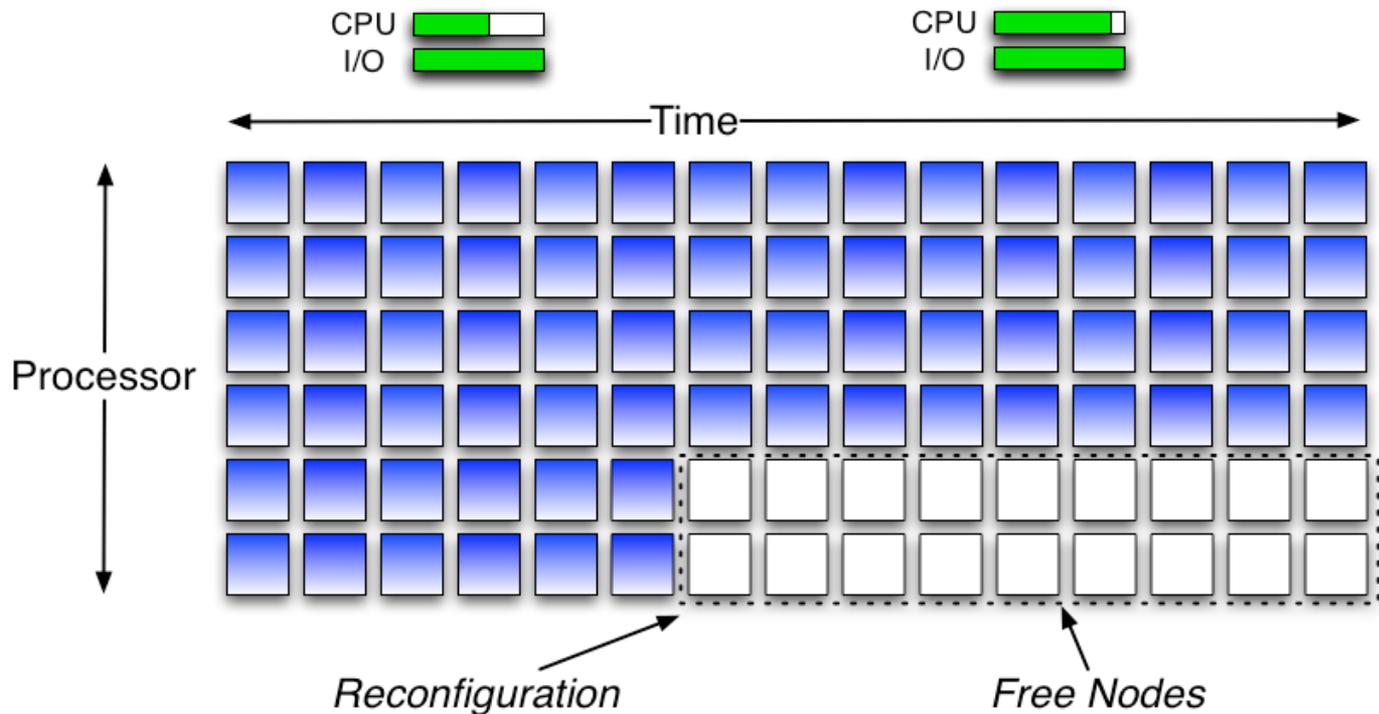
Congestion

- Resources are *congestible* if the use by one party has negative externalities on other users
 - Shared resources in HPC and file systems exhibit congestion, e.g. I/O throughput, IOPS, network, even memory
 - Congestion of one resource dominates performance
- Congestion pricing to mitigate congestion
 - Theory demands exponentially increasing price with utilization
 - Charge jobs to use congested resources
- Multi-resource optimization
 - Exploit elasticity in workloads
 - File systems asynchronous operation: trade memory for network, I/O
 - In HPC systems, we propose sharing non-congested resources



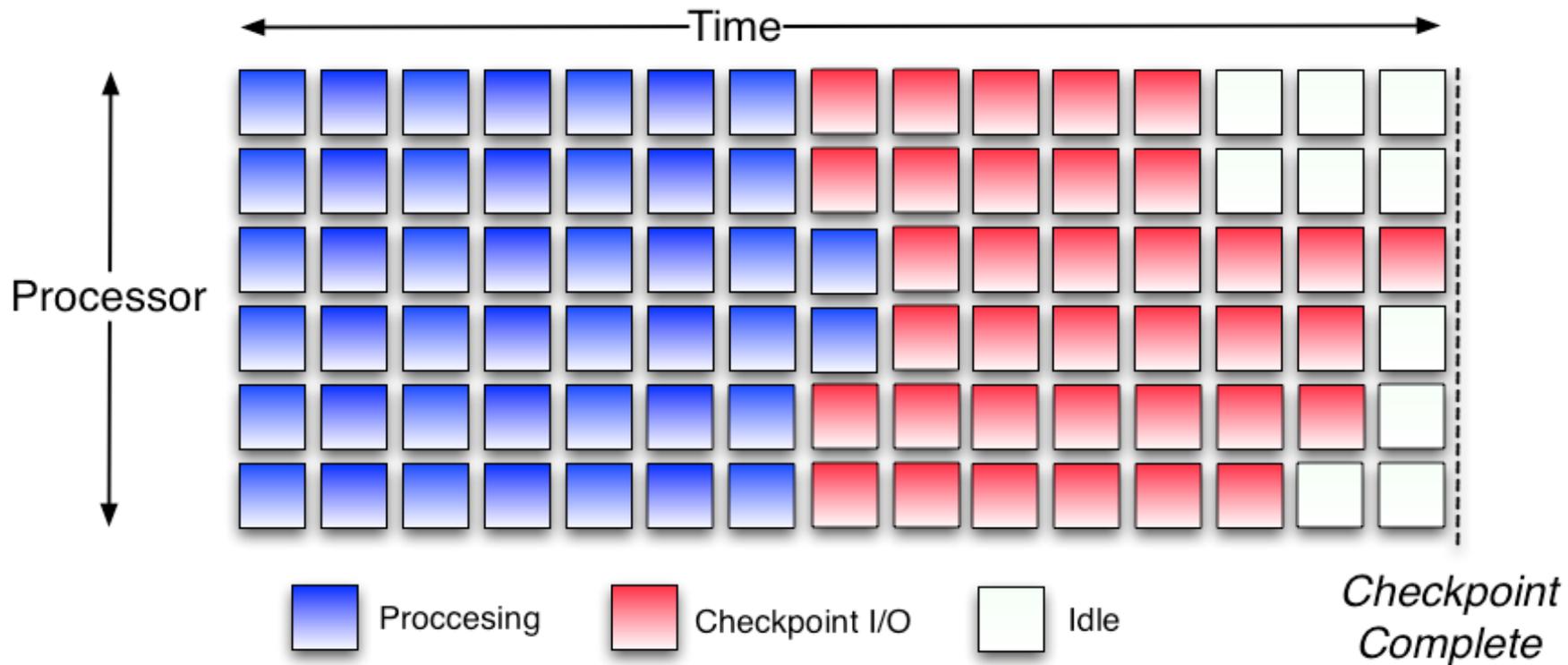
Scenario: *I/O Speed Matching*

- Code: Astronomy image reprocessing for visualization
 - Parallel data rate of 70GB/s keeps only 150 cluster nodes busy
- Reconfiguration balances utilization
 - Frees resources for other jobs



Scenario: *Checkpoint Stall*

- Code: n-body molecular dynamics
 - 1B particles on 100 nodes
 - Checkpoint takes about 20 seconds



What's broken?

Current allocation and scheduling practices

- Require users/programmers to understand the resource requirements of their code
 - Lead to gross over-provisioning
- Allocate entire nodes
 - Bundles all resources together
 - But, resources get used heterogeneously in time and space
- Does not manage I/O resources
 - Leads to congestion and stalls

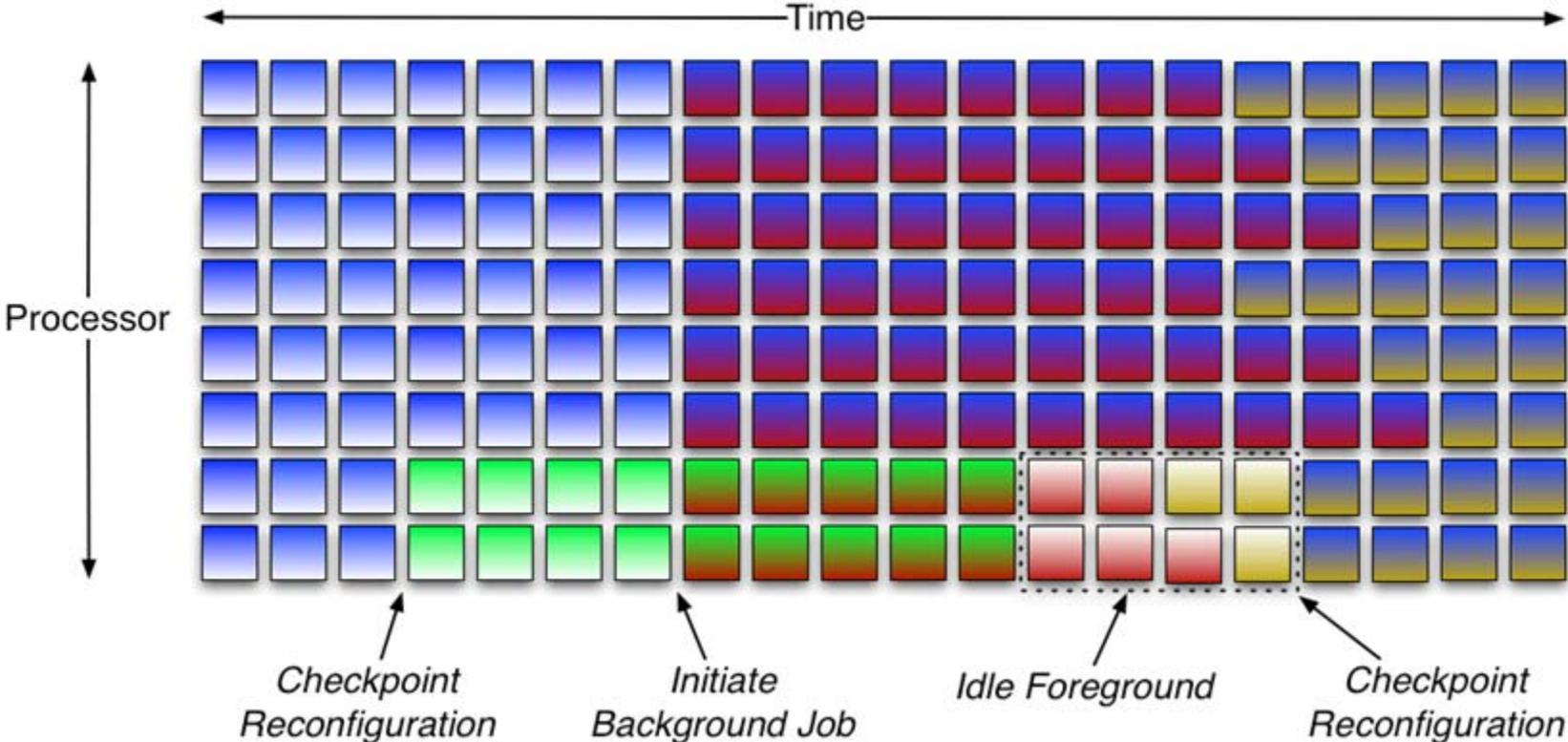
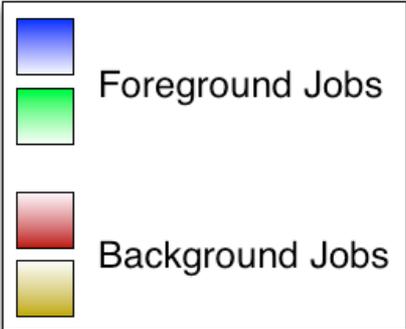


Mechanisms

- Reconfiguring parallelism
 - Balance consumption of I/O, network, memory, and processor
 - Restart after checkpoint on different number of nodes
 - Exploit spatial elasticity in HPC (number of nodes)
- Co-scheduling batch workloads
 - Shared resources with map/reduce, analysis, data mining, etc.
 - Exploit temporal elasticity (in resource consumption)
- Support low-priority *allocation elastic* jobs
 - Execute on arbitrary numbers of nodes/cores
 - Consume un-allocated resource fragments
 - Enables new classes of users (non-stakeholders)



CRAM Execution



Challenges

- Workload characterization
 - Determining how jobs will interact with resources and currently running jobs
 - Congestion makes profiling difficult or irrelevant
- Requirements on software
 - Parallel checkpoints
 - Tolerate reconfiguration
- Practicalities and realities
 - Performance interference for tightly coupled applications
 - H/W support for resource sharing



Education Mission

- Goal: to teach parallel programming as a core part of college and university curricula
- Disseminate JHU's *Parallel Programming* course
 - To schools without HPC facilities and multi-core labs
 - Programming exercises built on Web-services
 - Partners with regional (Mid-Atlantic) teaching colleges, community colleges, and historically-black colleges.
- Instructor workshops in 2010 and 2011
 - Instructor training
 - Customize curriculum for partner institution
 - Help prepare grants for cloud computing services

