True Elasticity in Multi-Tenant Data-Intensive Compute Clusters

Sriram Rao

CISL @ Microsoft

Oct. 15 2012

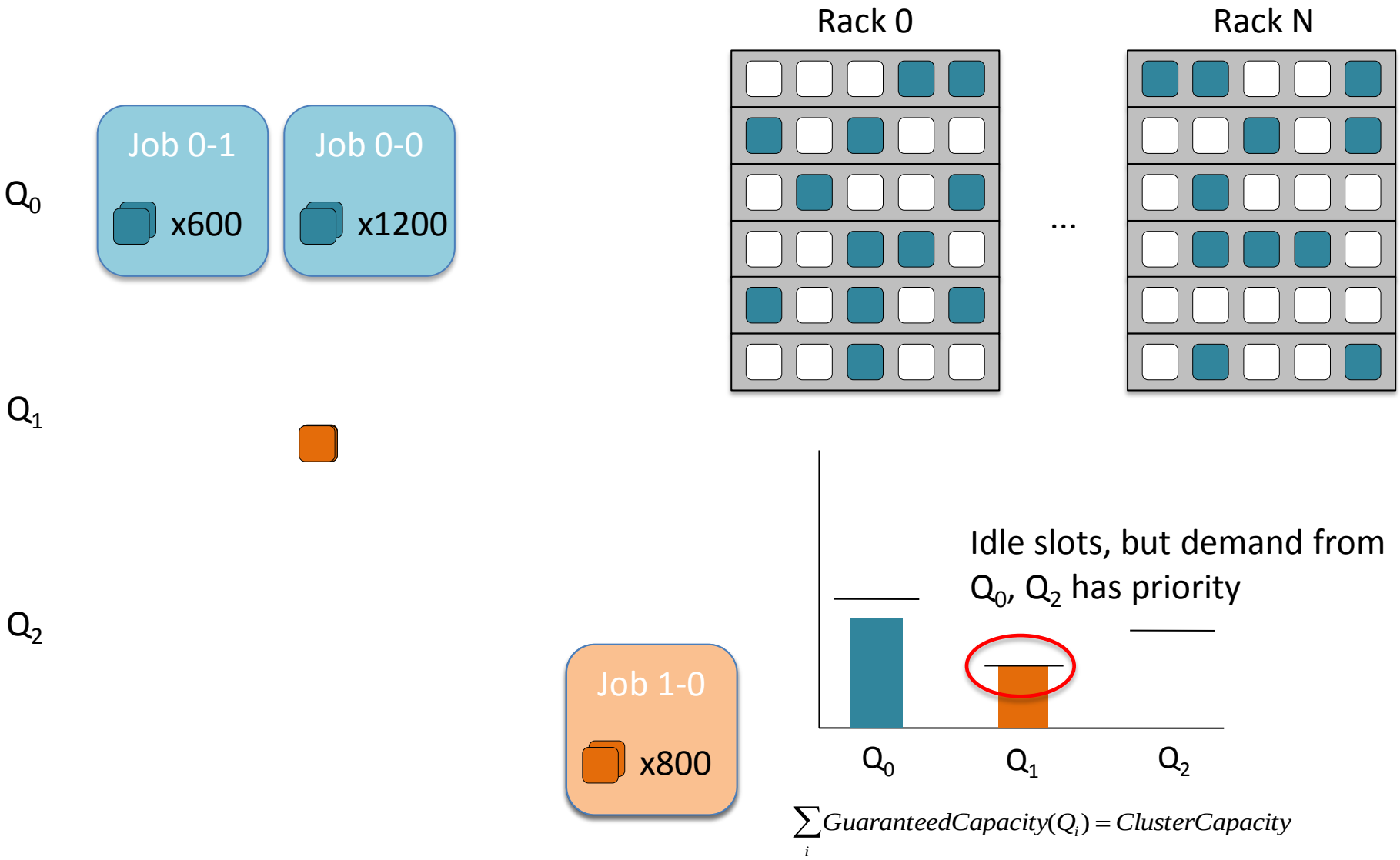
Joint Work With…

- Ganesh Ananthanarayanan, Ion Stoica (UC Berkeley)
- Chris Douglas, Raghu Ramakrishnan (Colleagues at Yahoo and now at Microsoft)
- Work started at Yahoo! Labs (and being continued in CISL@Microsoft)

Background

- “Big data” processing using compute frameworks in large clusters commonplace
  - Map-Reduce (Google), Cosmos (Bing), Hadoop (mostly everyone else)
- For economies of scale clusters are multi-tenanted
- Sharing of cluster resources (e.g., CPU) implemented via queues
  - Frameworks do not support preemption

Multi-tenanted Cluster



Idle Slots: To Use Or Not

- Compute frameworks do not support pre-emption
  - Task is the finest granularity for execution
- Sharing cluster resources:
  - In anticipation of future demand, leave idle slots fallow (Yahoo!)
    - Cluster utilization is ~70%
      - Tasks queued even though 30% of cluster is idle
  - Allocate idle slots to jobs, but kill tasks to enforce SLA's (Facebook, Bing)
    - Cluster utilization is 100%, but 21% tasks are killed

Problem Statement

- Compute frameworks force cluster operators to tradeoff utilization and efficiency:
  - Yahoo: 70% util, 100% efficiency
  - Bing: 100% util, 79% efficiency
- This matters…
  - In clusters of 1000's of nodes inability to use lots of nodes wastes $$$
  - Cloud provider: Margins are low and efficiency is paramount

Our Work

- *Amoeba*, a lightweight mechanism for enabling elasticity in data-intensive compute frameworks
  - Add preemption via a "checkpoint/restart" mechanism that saves task output and avoids wasting computation
- Resource consumption of jobs is elastic
  - Scale up/down usage based on cluster resource availability
- Preliminary results show that Amoeba can speed up jobs by 33%

Why Amoeba?

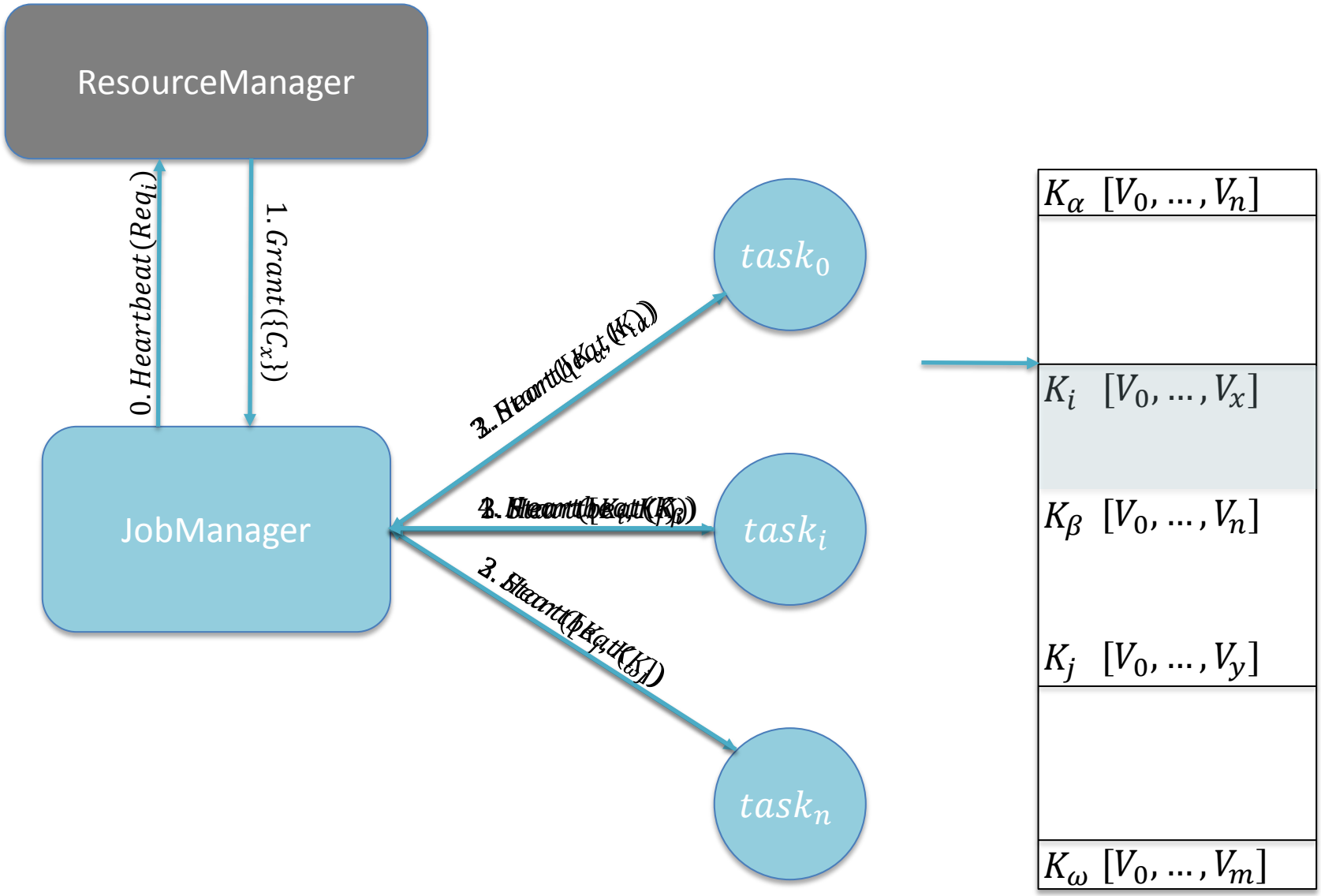
- Can't we just use "small" tasks?
  - Disk seeks, per-task scheduling overheads affect perf
- Can't we just use "uniform" size tasks?
  - Little correlation between task's input vs duration
- Can't we use OS preemption mechanisms?
  - Paging/swapping process state to disk is a no-no
    - Task heap sizes are in multiple GB's
    - Paging => Reboot the machine!

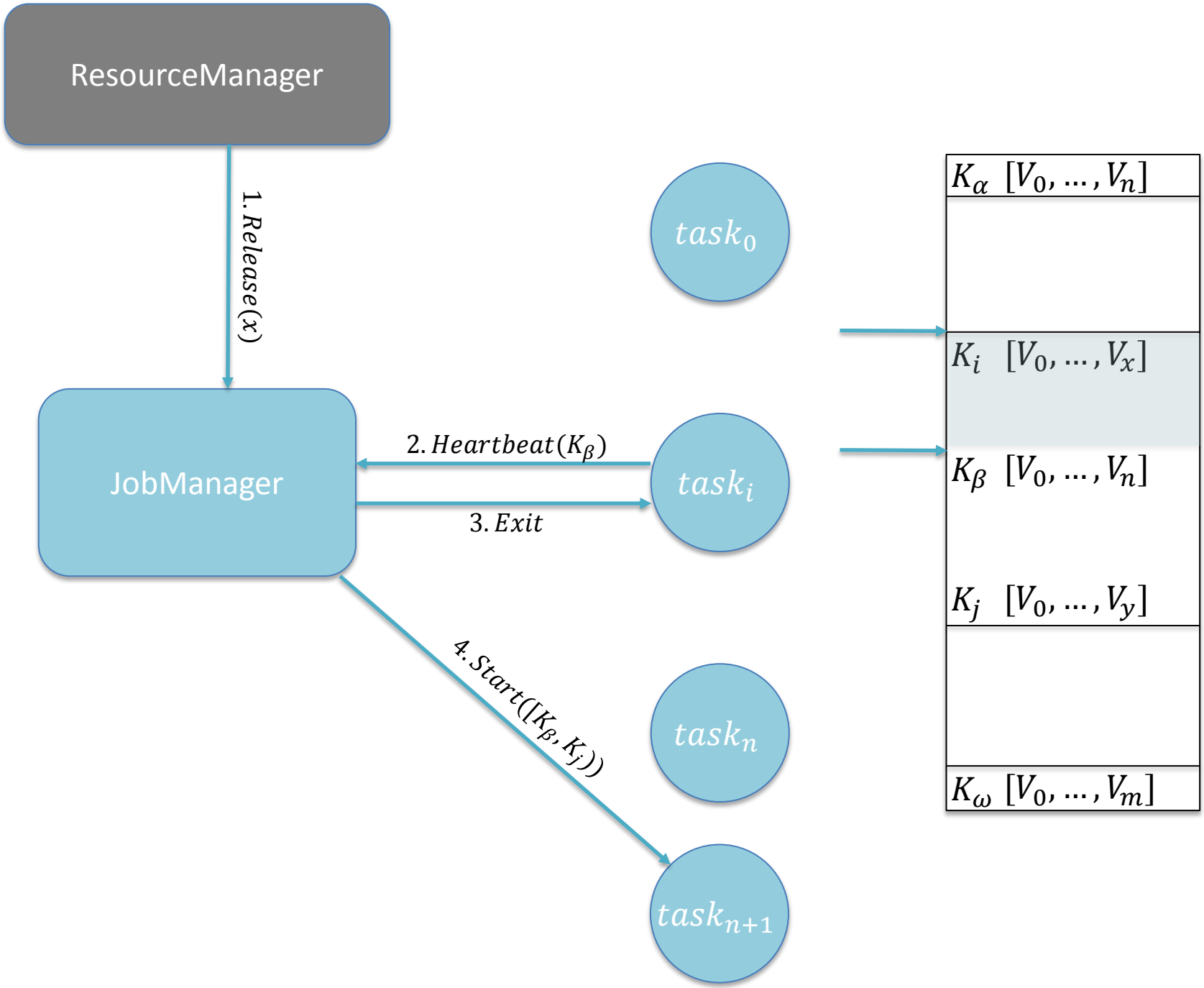
Amoeba Overview

- Idea: Construct checkpoint for task execution by identifying “safe” point
  - Terminate task execution at safe point
  - Spawn a new task for the remaining work
- Safe point provides mechanism for implementing preemption

Preemption for M/R

- [Dean'04] M/R programming model is based on keys
  - Map(k, v) ➔list(k', v')
  - Reduce(k, list(v)) ➔list(v')
- Safe point for an M/R task: Key boundary
  - Checkpoint task execution at a key boundary
    - Running task exits after saving state
  - Resume task execution at the next key
    - New task spawned for the remaining work
- Store key as memento of task execution
  - Lightweight mechanism

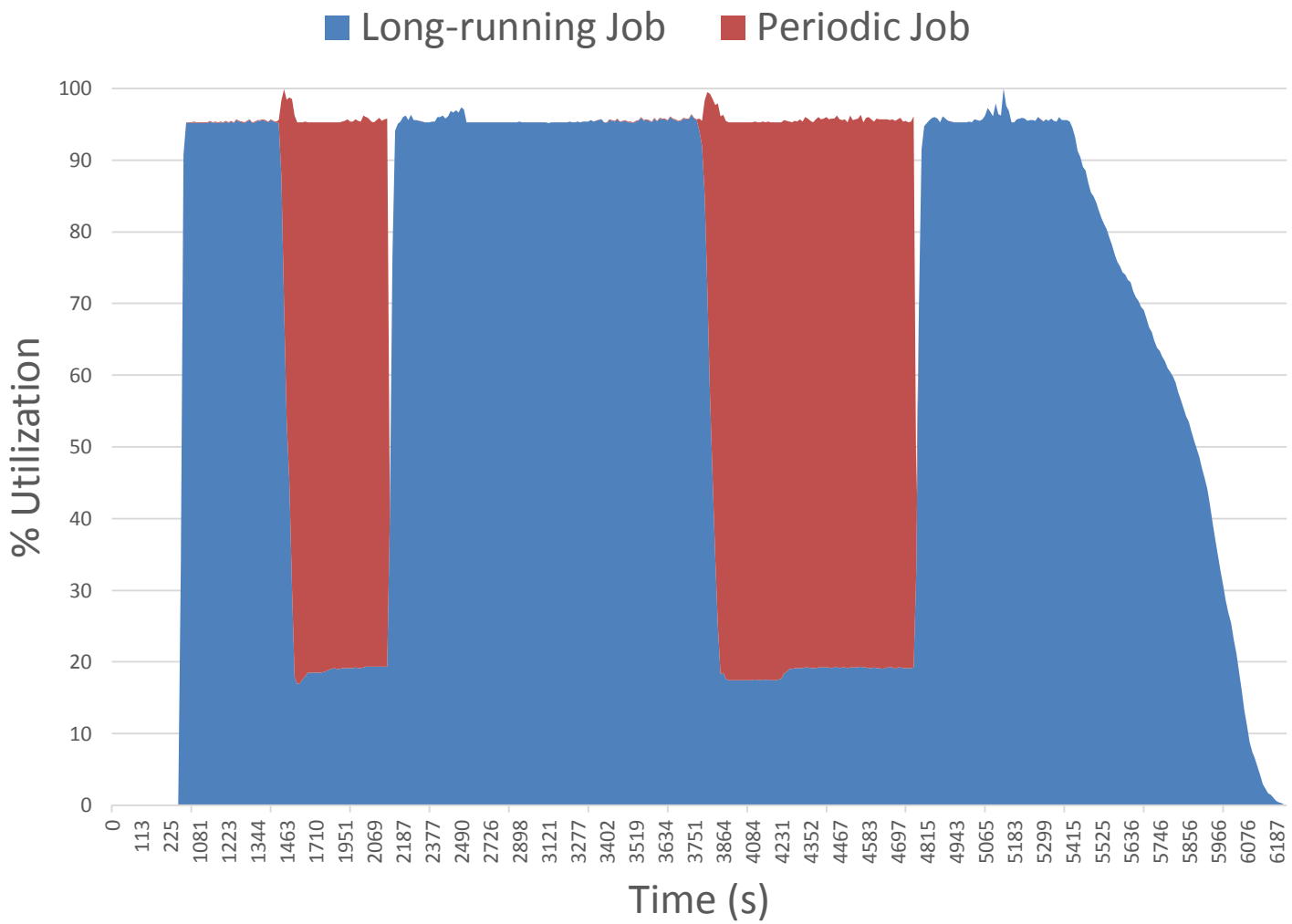

ResourceManager
$0.Heartbeat(Req_i)$
$1.Grant(\{C_x\})$
JobManager
$task_0$
$task_i$
$task_n$
K_α $[V_0, \ldots, V_n]$
K_i $[V_0, \ldots, V_x]$
K_β $[V_0, \ldots, V_n]$
K_j $[V_0, \ldots, V_y]$
K_ω $[V_0, \ldots, V_m]$


ResourceManager
1. Release(x)
JobManager
2. Heartbeat(K_β)
3. Exit
4. Start([K_β, K_j])
task_0
task_i
task_n
task_n+1
K_α [V_0, … , V_n]
K_i [V_0, … , V_x]
K_β [V_0, … , V_n]
K_j [V_0, … , V_y]
K_ω [V_0, … , V_m]

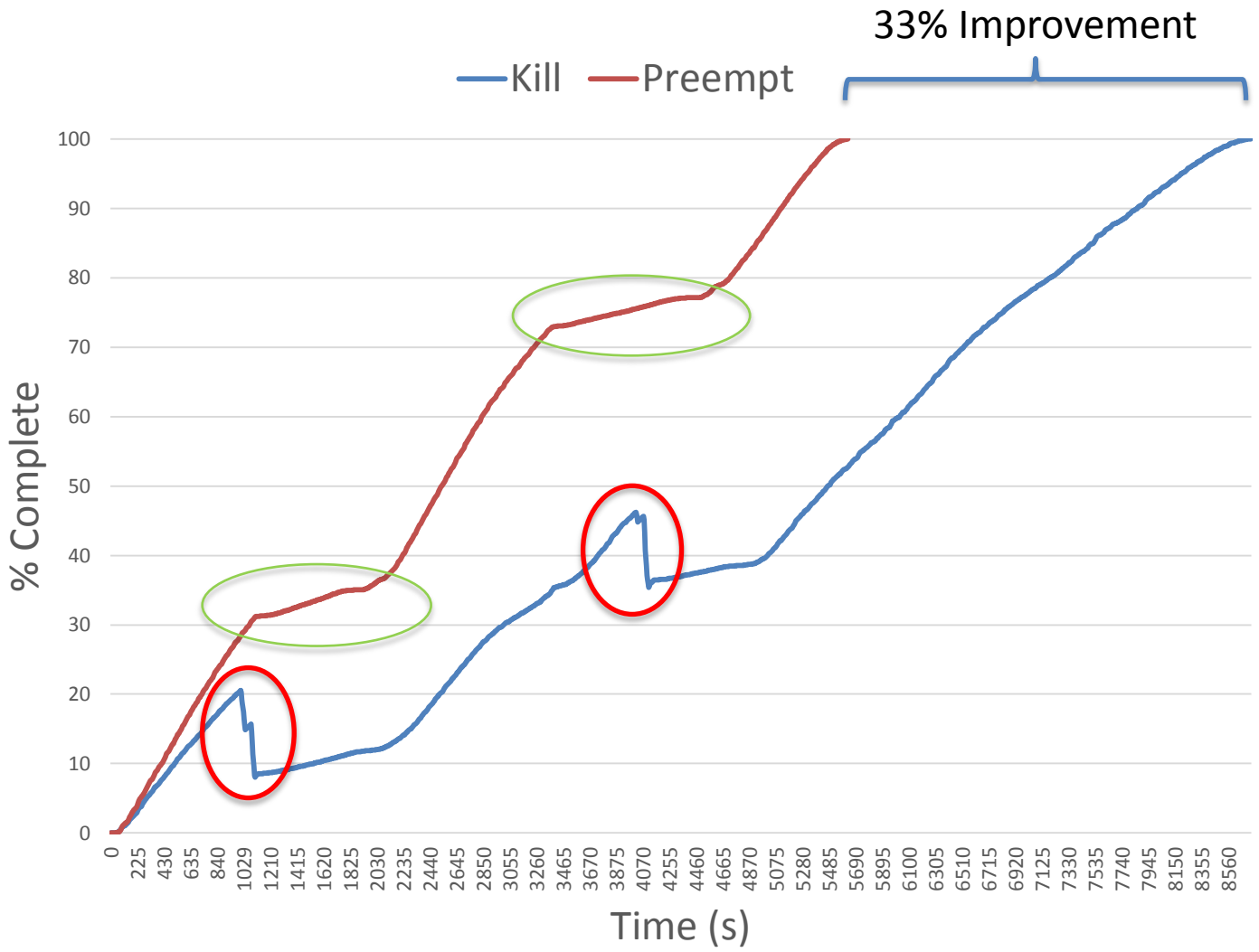
Amoeba Prototype

- Prototype implementation using Sailfish (SOCC'12)
  - Sailfish is based on Hadoop 1.x
  - Prototype is released to open source
    - http://code.google.com/p/sailfish
- Preemption is simplistic
  - When a job manager is asked to release a slot, it chooses a task arbitrarily

Elasticity With Amoeba



Killing Tasks vs Preemption



- Idea seems promising
- What does Amoeba enable?

Preemption <-> Scheduling

- Scheduling algos for compute frameworks do not factor in pre-emption
  - Slots are allocated to tasks and are held until task completes
- Amoeba makes it possible to modify scheduling decisions
  - Migrate a running task to enable locality for a new task
  - “De-frag” a cluster by migrating tasks to create “bigger slots
- Scheduling in the context of YARN is on-going work

(Some) Applications of Elasticity

- Handle (computation) skew
  - If task is taking too long to finish, checkpoint and then spawn multiple tasks to handle the remaining work
- Improve efficiency of speculative execution
  - Now: Launch multiple tasks and pick the winner
  - .Next: Adjust end-point of running task and launch new task for what is left
- In cloud settings, elasticity enables effective use of “spot instances”
  - Checkpoint work done by tasks running on spot instances

On-going Work

On-going Work

- Build Amoeba in the context of YARN (Hadoop 2.x)
  - http://issues.apache.org/jira/browse/MAPREDUCE-4584
  - http://issues.apache.org/jira/browse/YARN-45
- Work started at Yahoo labs, is now being continued in CISL at Microsoft
  - We intend to release our work to open source