Middleware and Virtualization Management

Resource Co-Allocation for Large-Scale Distributed Environments

Claris Castillo (IBM T.J. Watson Research Center)
George Rouskas (North Carolina State University)
Khaled Harfoush (North Carolina State University)

International ACM Symposium in High Performance Distributed Computing

June 13, 2009

Motivation (1)

- **Co-allocation of resources: allocation of multiple resources within the same time window**
- **Emergence of new paradigms**
  - On Demand computing (Amazon EC2, SalesForce, IBMCloud)
- **Requirements**
  - QoS/SLA support
  - Efficiency
  - Scalability
- **Emergence of new applications that capitalized on the availability of distributed computing to perform tasks with spatial and temporal dependencies (MapReduce/financial apps.)**

Motivation (2)

- **Applications**
  - Virtual Computing Lab (VCL)
  - MapReduce framework (Hadoop)
  - Grid lambda scheduling
  - Workflow scheduling

Background

- **Naïve Approach: A co-allocation request can be treated as a group of sequential scheduling requests**
  - Inappropriate for time sensitive applications
- **Batch scheduling**
  - Resource driven: optimizing for system performance
    - Limited support for QoS by means of backfilling and priorities
  - Job driven: optimizing for application performance
- **Advance reservations**
  - QoS provisioning
  - Workflow support
  - Multiple drawbacks

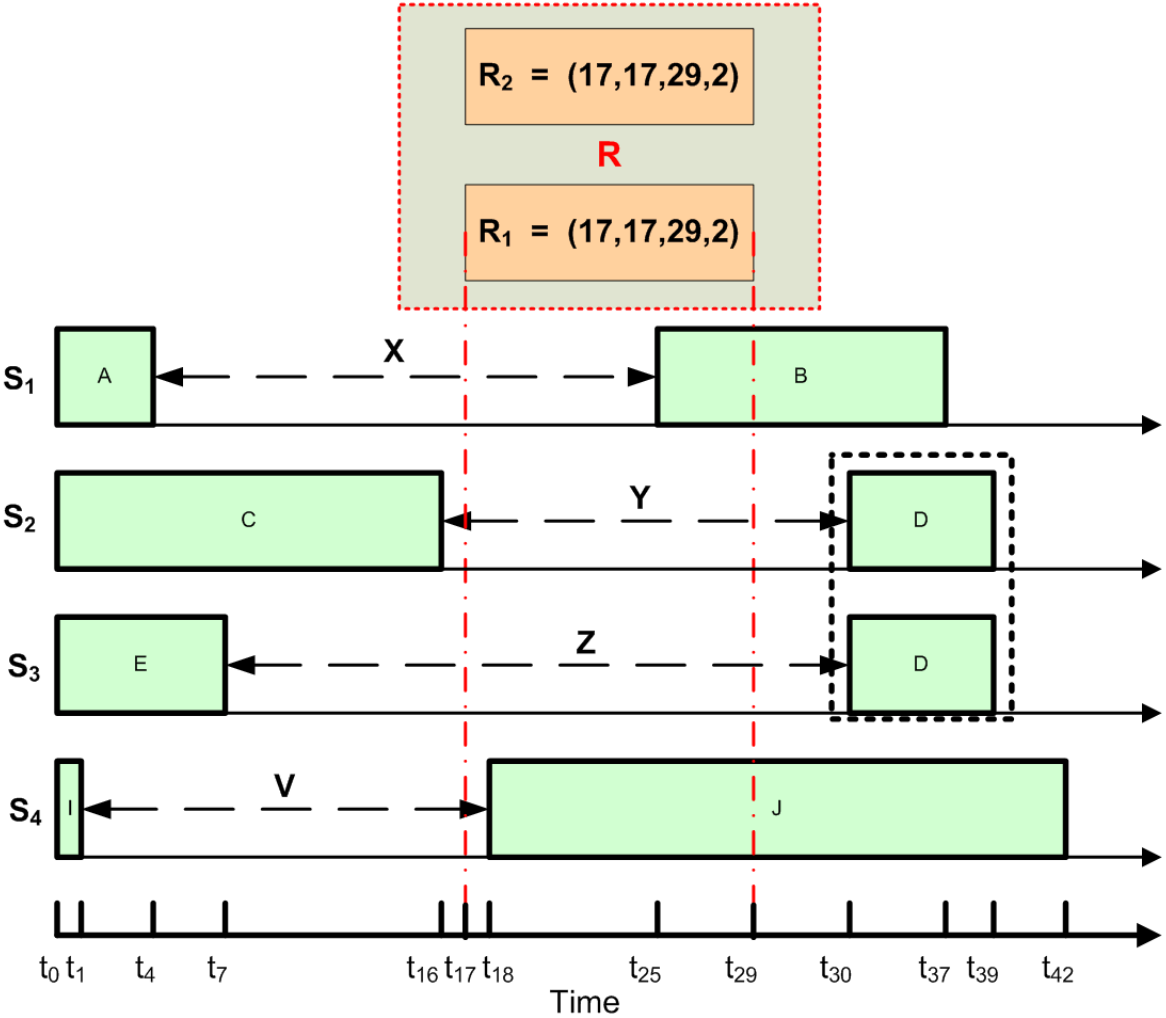
Goals

- **Providing users with time guarantees by scheduling jobs as they arrive** *without promoting resource fragmentation*
- **Allowing better scheduling decisions by keeping look ahead until the horizon of the schedule in a way that is** *efficient*

Contributions

- **A co-allocation scheduling algorithm**
  - Effective in co-allocating resources and provides support for *advance reservations* and *range search*
- **Range search**
  - Ability of the system to find a set of resources available within a given time window
  - Enable selection and scheduling algorithms that are application specific
- **Efficient data structure to organize resource availability**
  - Leading to the design of an algorithm that allows a single search operation to identify all required resources *efficiently*

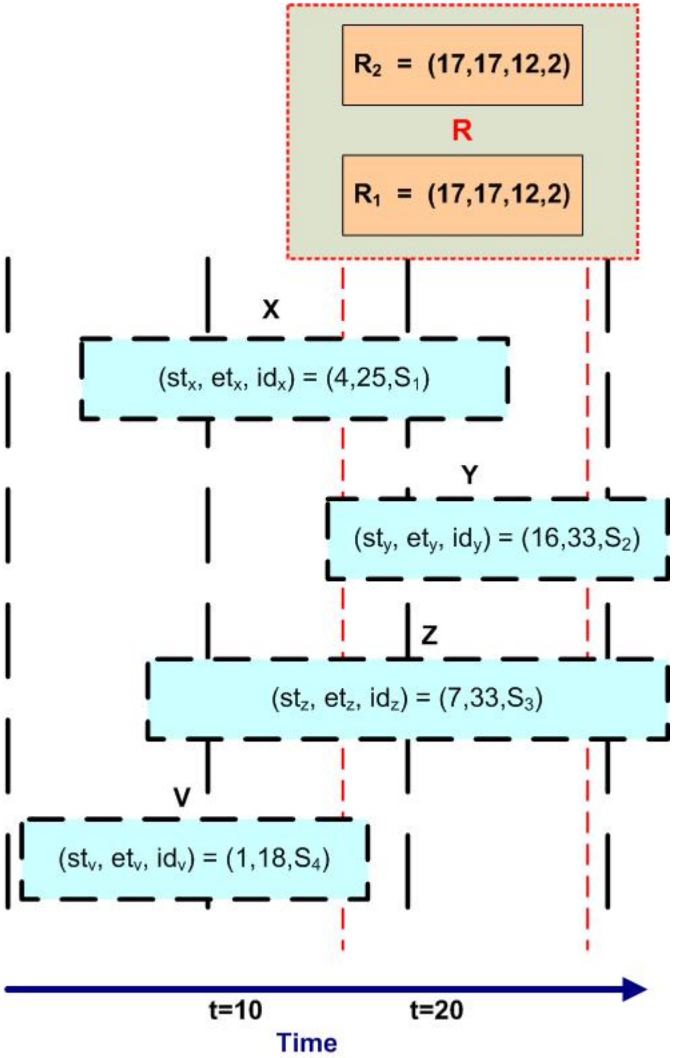
Problem Description



System Model

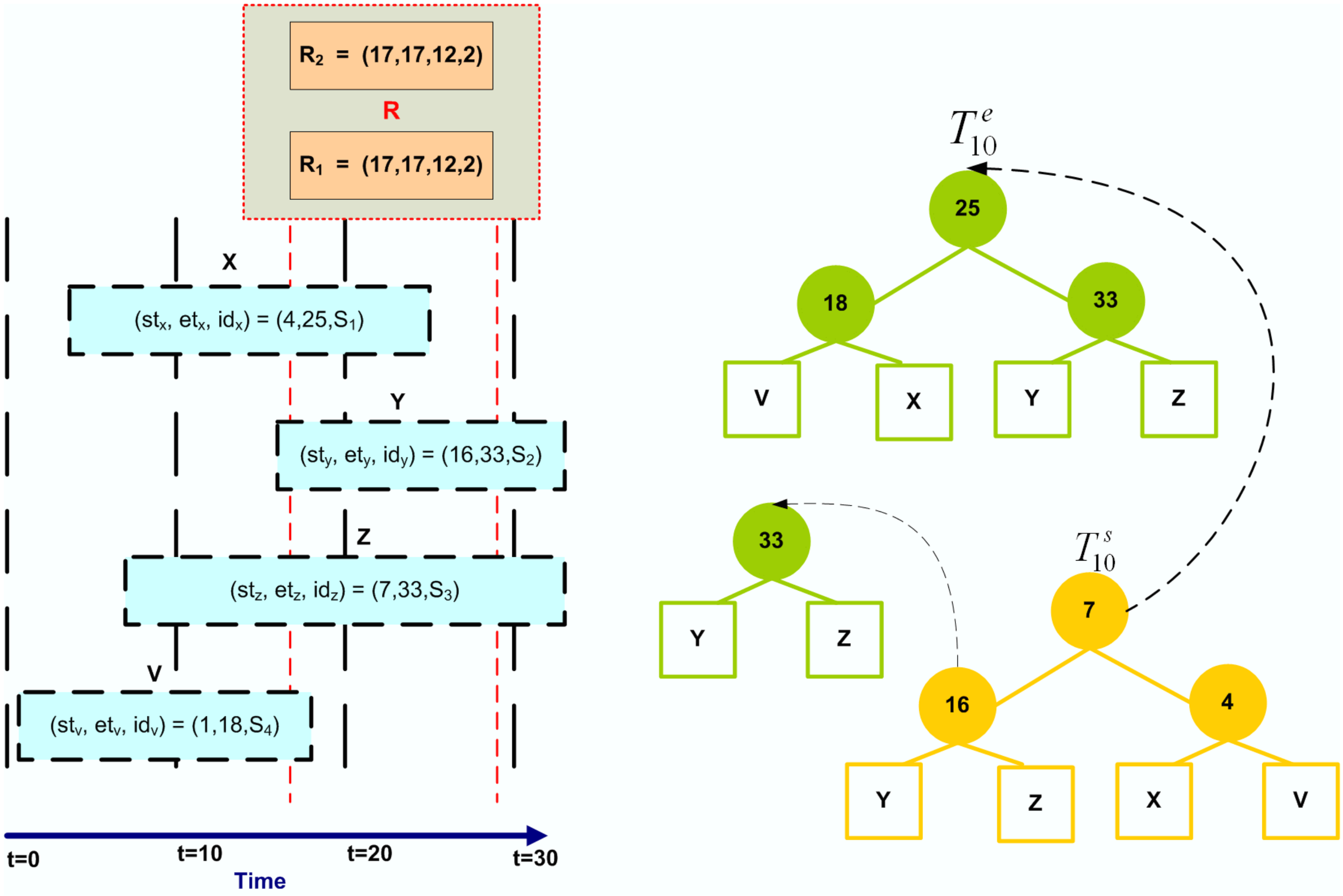
- **We consider the following settings:**
  - Scheduler ***S***
  - ***N*** servers
  - Reservation request ***r*** requires service
  - Request (q_r, s_r, l_r, n_r)
    - q_r request time
    - s_r earliest time the reservation is needed
    - l_r temporal size of the request (duration)
    - n_r spatial size of the request (no. or servers)
  - Idle period (st_i, et_i, id_i)
    - st_i starting time
    - et_i ending time
    - id_i server offering the idle period

Data Structure and Algorithm



- **Time space is partitioned into time slots of equal length**
- **Idle periods are stored in each time slot they span over**
- **Algorithms searches only into the time slot containing s_r**
- **Upon failure to schedule: $s_r = s_r + \Delta_t$**
- **Honor atomicity of the request by means of temporal counters**
- **Number of idle periods per time slot can be bounded to N if time slot size is set to the minimum temporal size**

Data Structure



Two feasibility criterion: **$\underline{st_i < st_x}$** and **$\underline{et_i > et_x}$**

Performance Evaluation

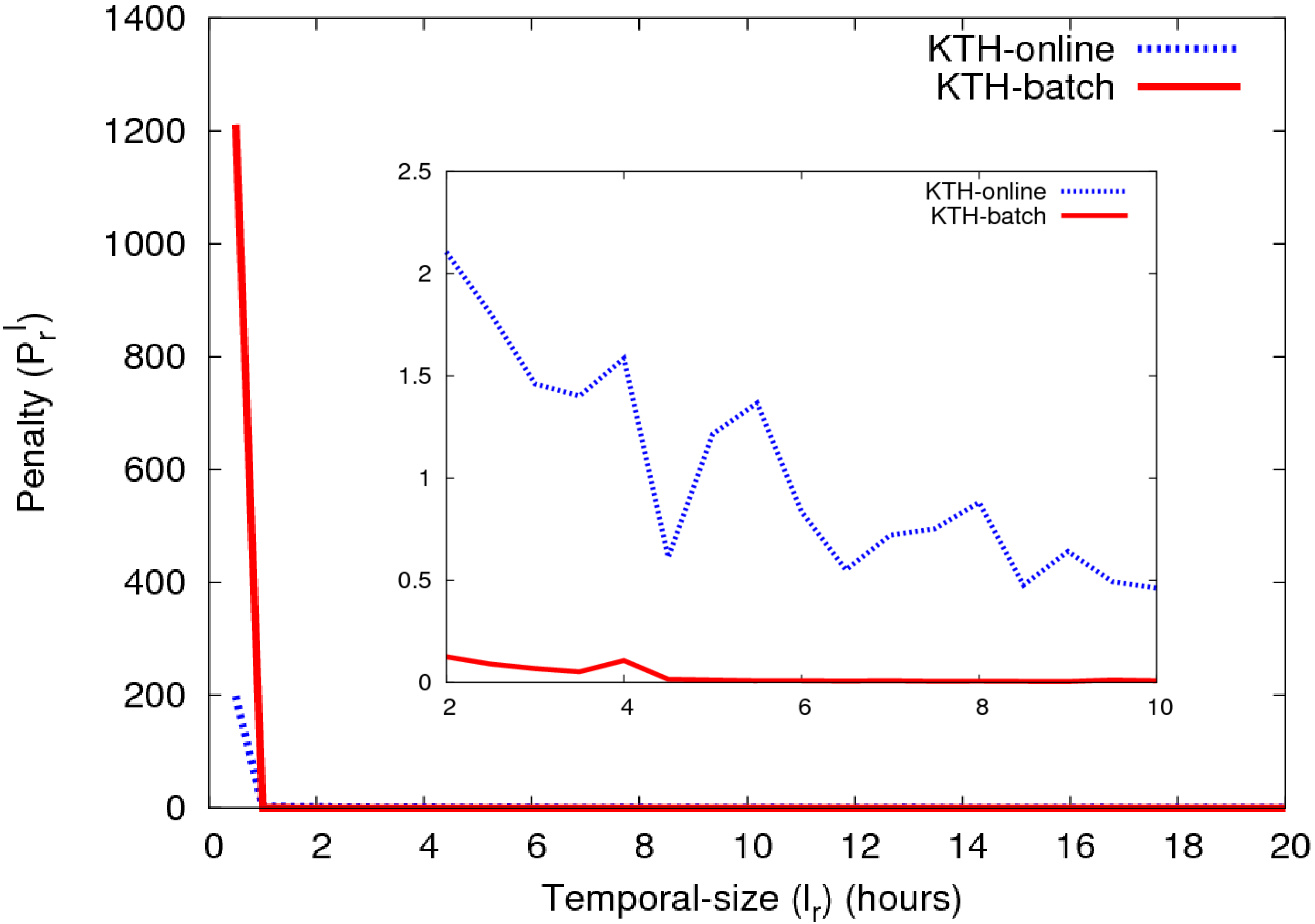
- **Real workloads drive simulations [ParArch]**

Workload	No. of processors	No. of jobs	Avge. length (hrs)	Avge. spatial size
CTC	512	39,734	5.82	9.48
KTH	128	28,481	2.46	7.67
HPC2N	240	202,825	4.72	6.56

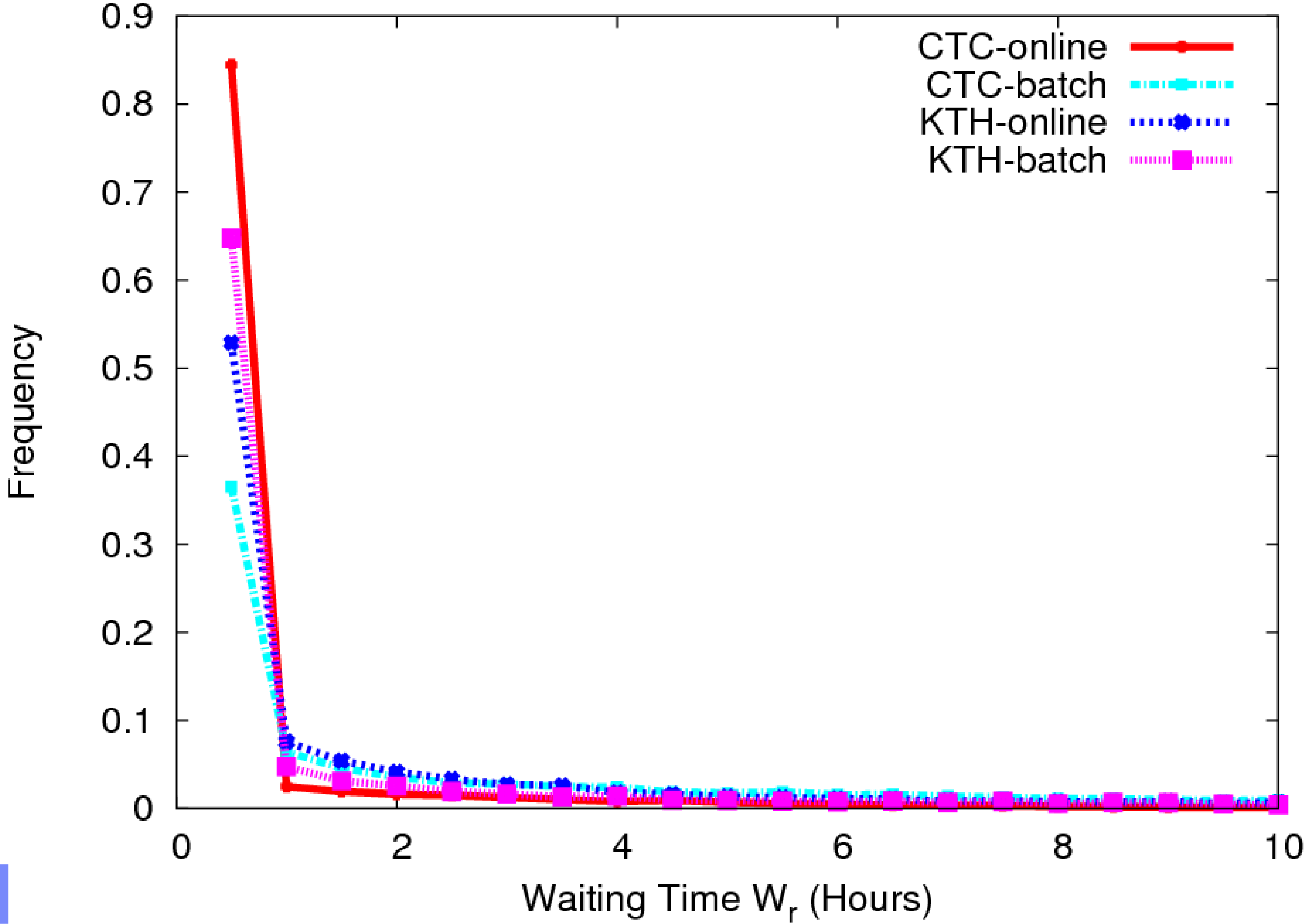
- Two experiments
  - Comparison to batch scheduling
  - Impact on performance of advance advance-reservations

[ParArch] Parallel Worklaod Archive at www.cs.huji,ac.il/labs/parallel/workload/ .

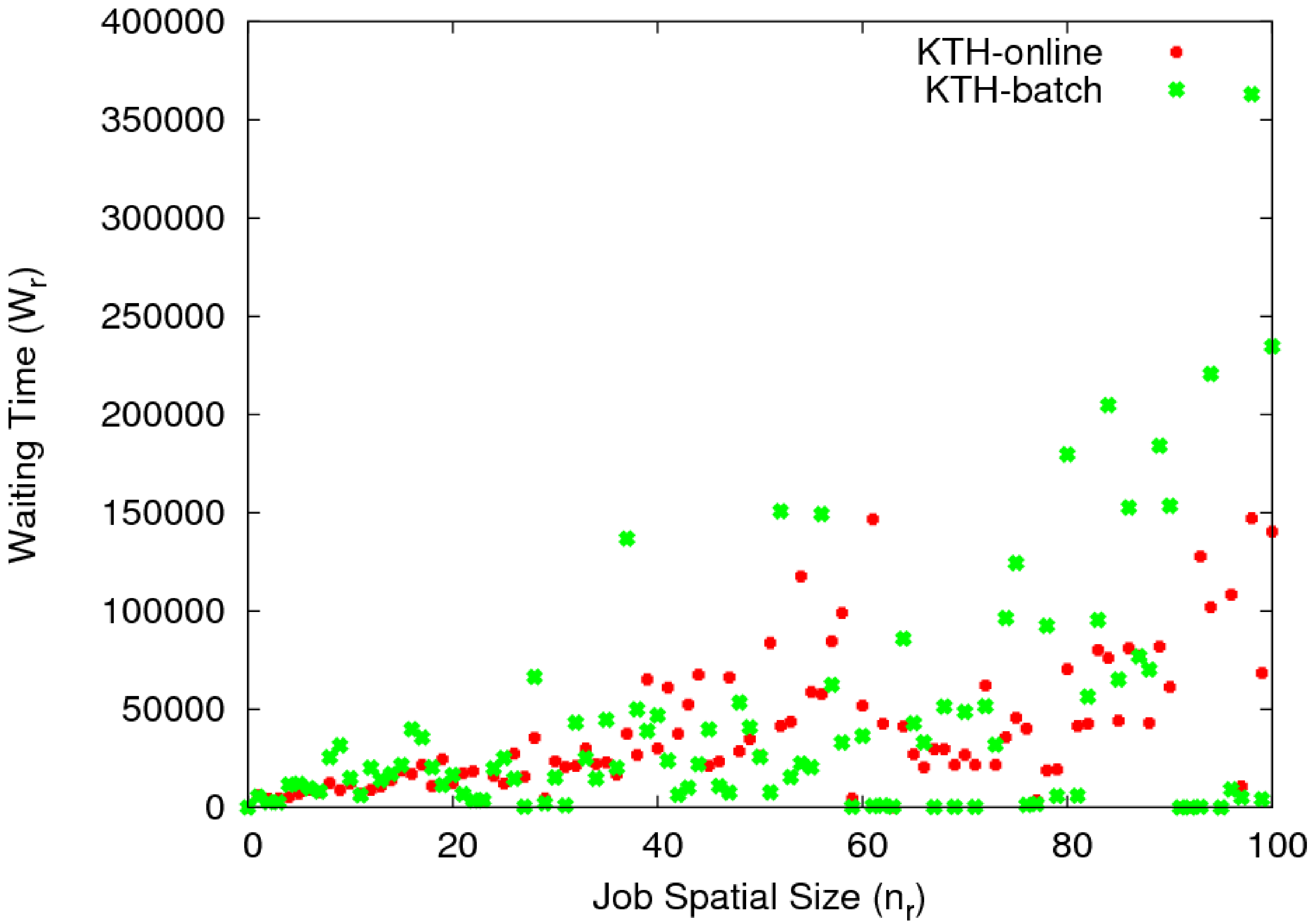
Temporal-size Penalty



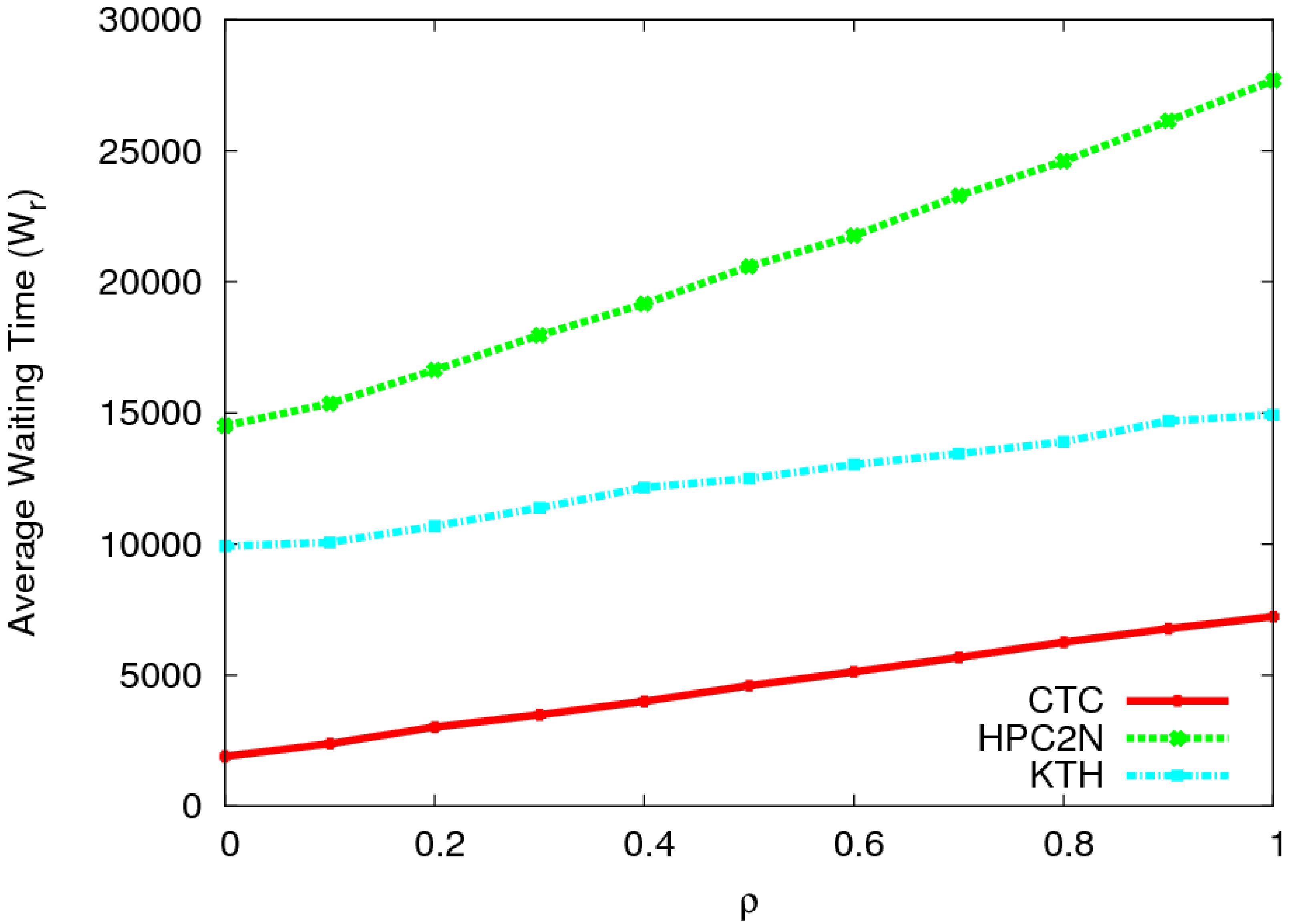
Waiting Time Distribution



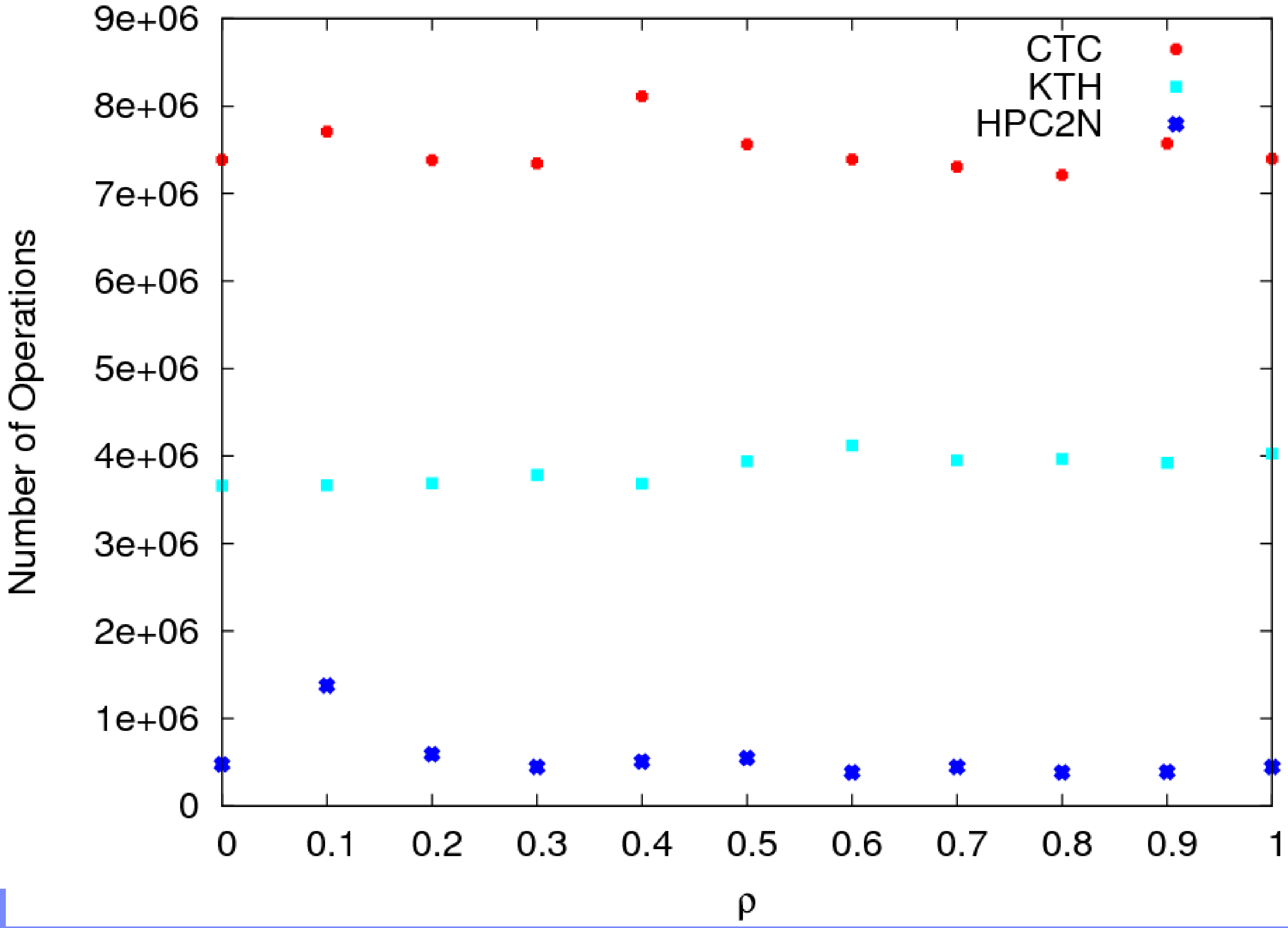
Waiting time distribution as a function of spatial size



Avg Waiting Time vs. fraction of advance reservations (ρ)



Number of operations vs. fraction of advance reservations (ρ)



Number of retrials vs. spatial size

Workload/n	(0:50]	(50:100]	(100:150]	(150:200]	(350:400]
CTC (No. of retrials)	2.96	5.34	7.22	13.25	127.44
KTH (No. of retrials)	10.27	60	120	--	--

- Larger spatial size distribution results in larger number of attempts.
- Temporal size distribution of KTH shows large proportion of small jobs.

Discussion of Results

- **Our algorithm can efficiently co-allocate resources while supporting advance reservations**
- **Online advance reservations mechanisms might offer a better solution to the problem of co-allocating resources as compared to conventional batch scheduling**
- **Our work can be easily extended to support deadlines**

Future Work

- **Implement the co-allocation algorithm proposed in the context of**
  - Hadoop
  - End-to-end path problem in Grid lambda scheduling
- **Impact of workload characteristics on system/user performance**
- **Uncertainty of completion times**

Thank you!

I can be reached by email at:

claris@us.ibm.com