Managing Descheduling Risk in the Google Cloud

Walfredo Cirne, Geeta Chaudhry, Scott Johnson

The Problem

- How can we be sure that we have enough spare resources to restart tasks that die due to hardware failure ...
  - … while minimizing the resources we need to set aside
  - … and considering correlated failures

Google

Google Cloud Structure

- Datacenter
  - The actual building we keep our servers
- Cluster
  - A set of computers physically colocated within a datacenter, sharing the same local network
- Cell
  - A fraction of the cluster scheduled by the cell manager

Using a cell

- Users submits a job, composed of multiple tasks
- Each task must specify how much memory, CPU, network, disk it needs
- Tasks may also specify constraints on the machines they run
- Jobs may have constraints on how its tasks are distributed
  - For example, no more than 5 tasks per rack

- Machines fail or otherwise become unanavailable (e.g. kernel update)
  - Failures may be correlated (e.g. rack)
- Jobs are written to tolerate task restart (within a reasonable rate)
- We say a task deschedule if we cannot restart it

Admission control for a cell

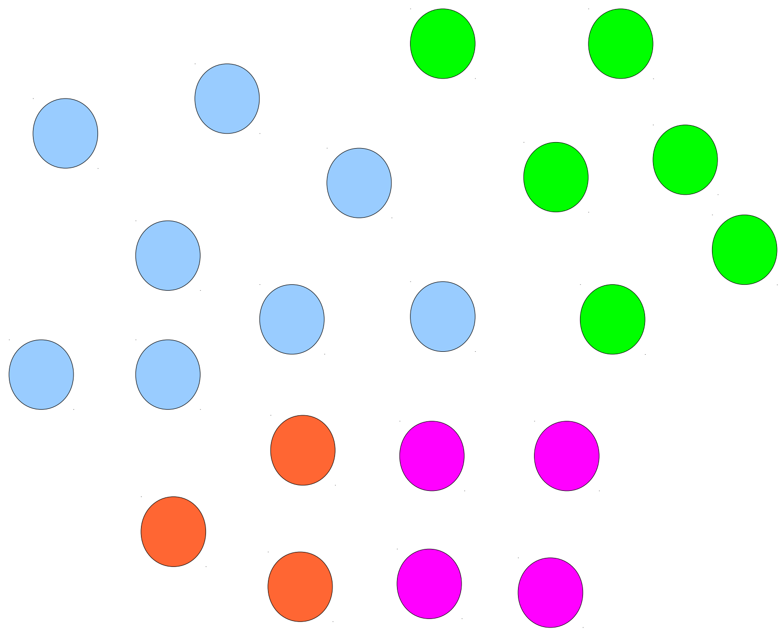
- We would like to ensure that task descheduling never happens
- We do so by setting up admission control for jobs
- We only admit jobs when we are very confident we can restart its tasks
- But, how can we achieve this confidence?
- How can we reserve (the minimal amount of) resources to ensure we can withstand a given failure rate?

Backup tasks

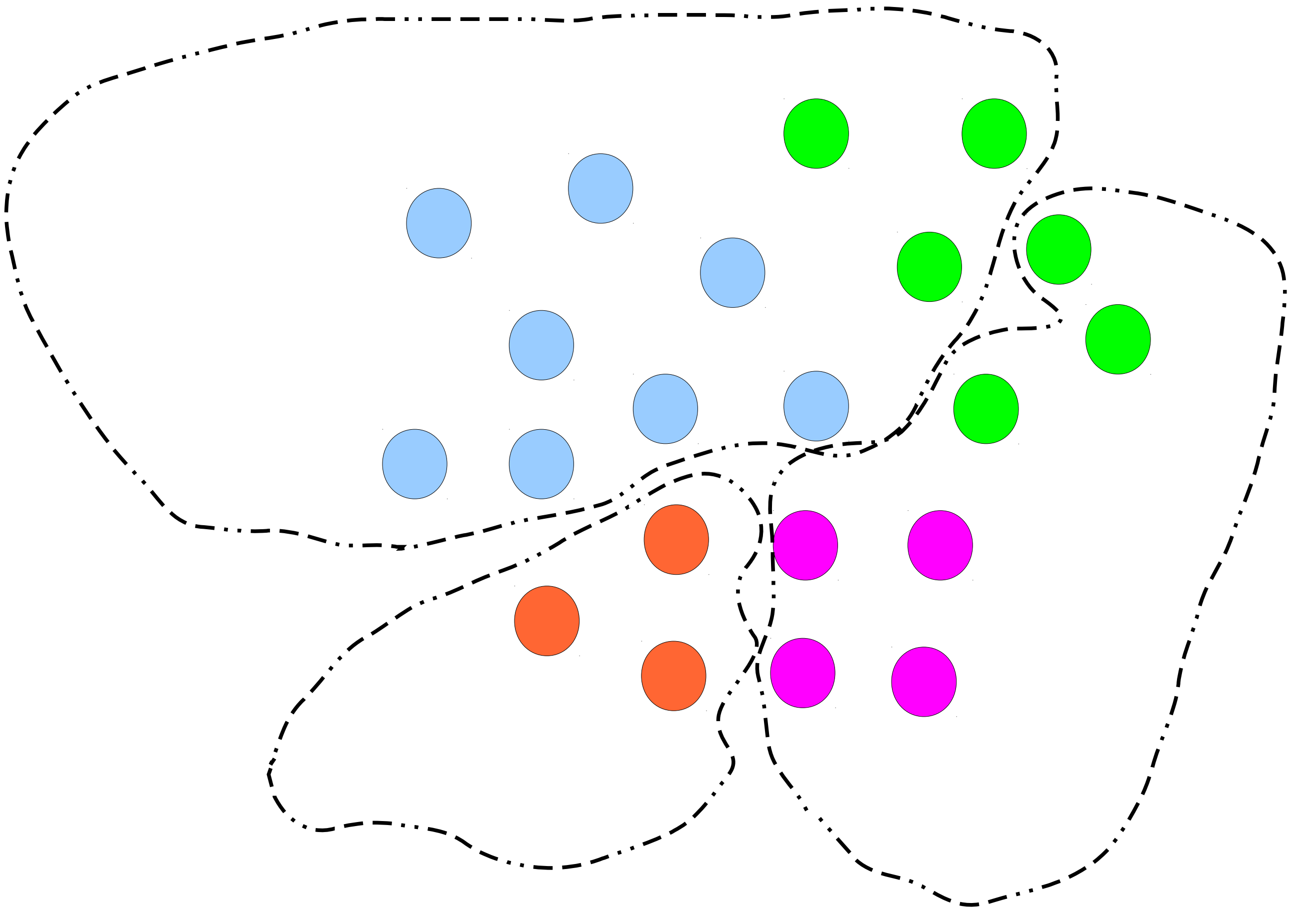
- We can precisely compute the probability distribution function that **x** tasks deschedule in a bag of given **n** tasks
  - A bag of tasks is just a set of tasks that is invisible to the user
- This ability enables us to do active risk management
  - For a given bag of **v** tasks, add **x** backup tasks to the bag, such that the probability that no more than **x** failures out of the total **n** = **v** + **x** **tasks** is less than a chosen threshold
  - A backup task is one that can replace any tasks in the bag
  - For example, if [ram = 1GB, cpu = 1.0] and [ram = 10MB, cpu = 2.0, attrib = value] are in the bag, a backup task must at least have [ram = 1GB, cpu = 2.0, attrib = value]

Google

Tasks

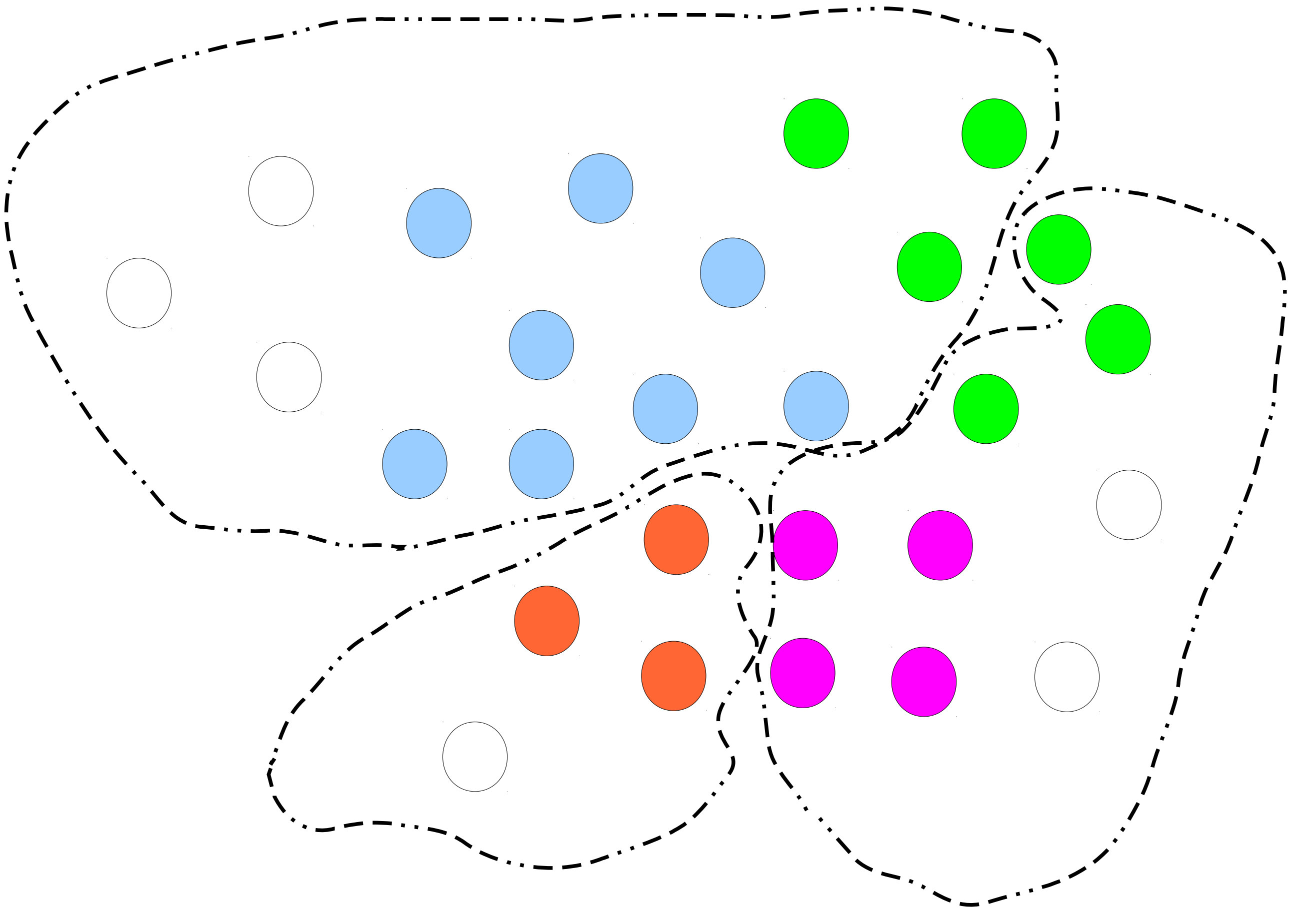


Task bags



Google

Adding backup tasks



Probability of **x** tasks failing in a bag

- We want the probability distribution P(**f** = **x**), where **f** is the number of tasks to deschedule in a given task bag **B**
- Let's start by assuming that:
  - No more than one task of bag **B** run on the same machine
  - All tasks of bag **B** run on a given rack
- We'll remove these assumptions later

Failure probability for a rack

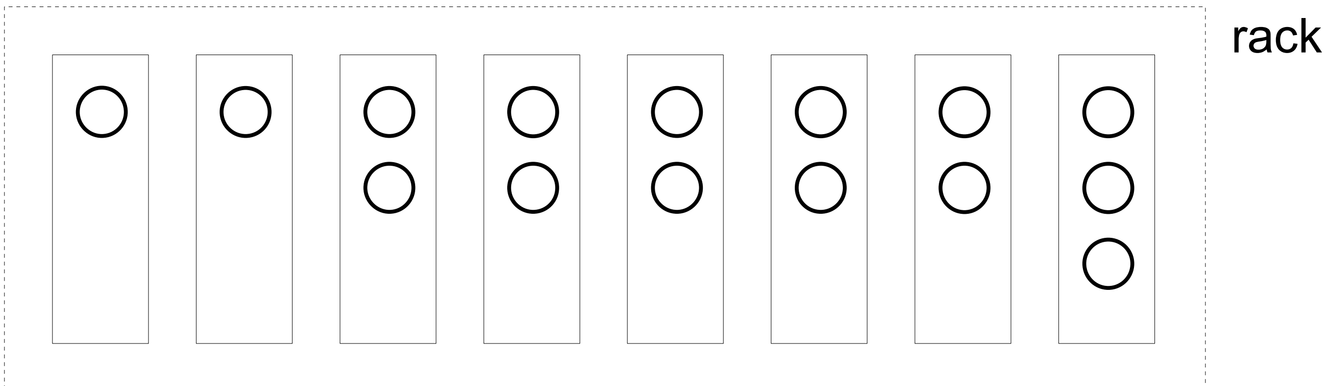
- Let P**r**(**f** = **x**) be the probability that **x** tasks deschedule on rack **r**
- Let p(**r**) be the probability that rack **r** fails
- Let p(**m**|**~r**) be the probability the machine **m** fails but the rack **r** that has **m** has not
- Let **R** be the number of machines used by bag **B** in rack **r**
- We want to compute P**r**(**f** = **x**) from p(**r**), p(**m**|**~r**), and **R**

- P**r**(**f** > **R**) = 0
- P**r**(**f** = **R**) = p(**r**) + p(**~r**)·PI**r**(**f** = **R**)
- Pr(**f** = **x** < **R**) = p(**~r**)·PI**r**(**f** =**x**)
- PI**r**(**f** = **x**) = Binomial(**x**, **R**, p(**m**|**~r**))

Google

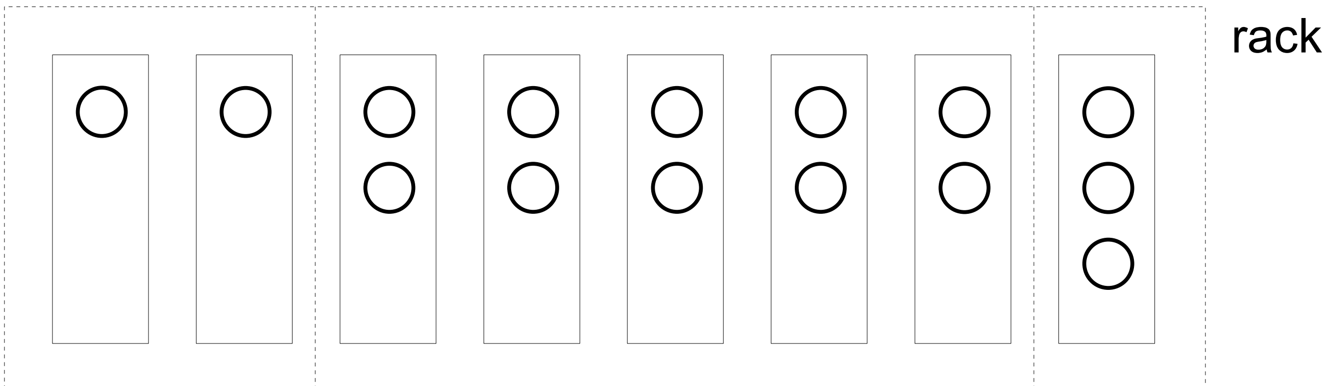
Two or more tasks in the same machine

- Change the computation PI**r**(**f** = **x**) to consider that each machine failing will bring down **i** tasks



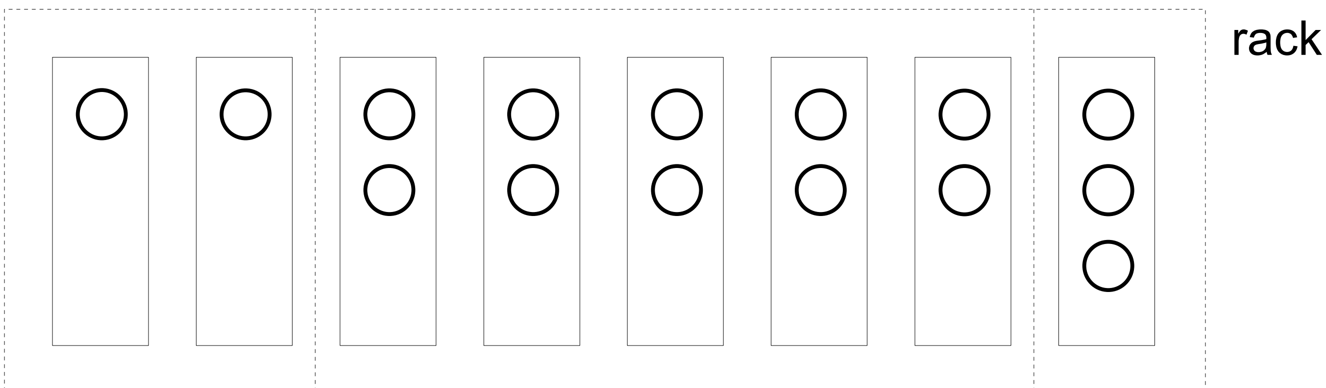
Two or more tasks in the same machine

- Change the computation PI**r**(**f** = **x**) to consider that each machine failing will bring down **i** tasks



Two or more tasks in the same machine

- Change the computation $\mathrm{PI}\mathbf{r}(\mathbf{f} = \mathbf{x})$ to consider that each machine failing will bring down **i** tasks



$\mathbf{i} = 1$

$\mathrm{PI}\mathbf{r}_1(\mathbf{f} = 0)$

$\mathrm{PI}\mathbf{r}_1(\mathbf{f} = 1)$

$\mathrm{PI}\mathbf{r}_1(\mathbf{f} = 2)$

$\mathbf{i} = 2$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 0)$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 2)$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 4)$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 6)$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 8)$

$\mathrm{PI}\mathbf{r}_2(\mathbf{f} = 10)$

$\mathbf{i} = 3$

$\mathrm{PI}\mathbf{r}_3(\mathbf{f} = 0)$

$\mathrm{PI}\mathbf{r}_3(\mathbf{f} = 3)$

Probability of **x** tasks failing in a bag

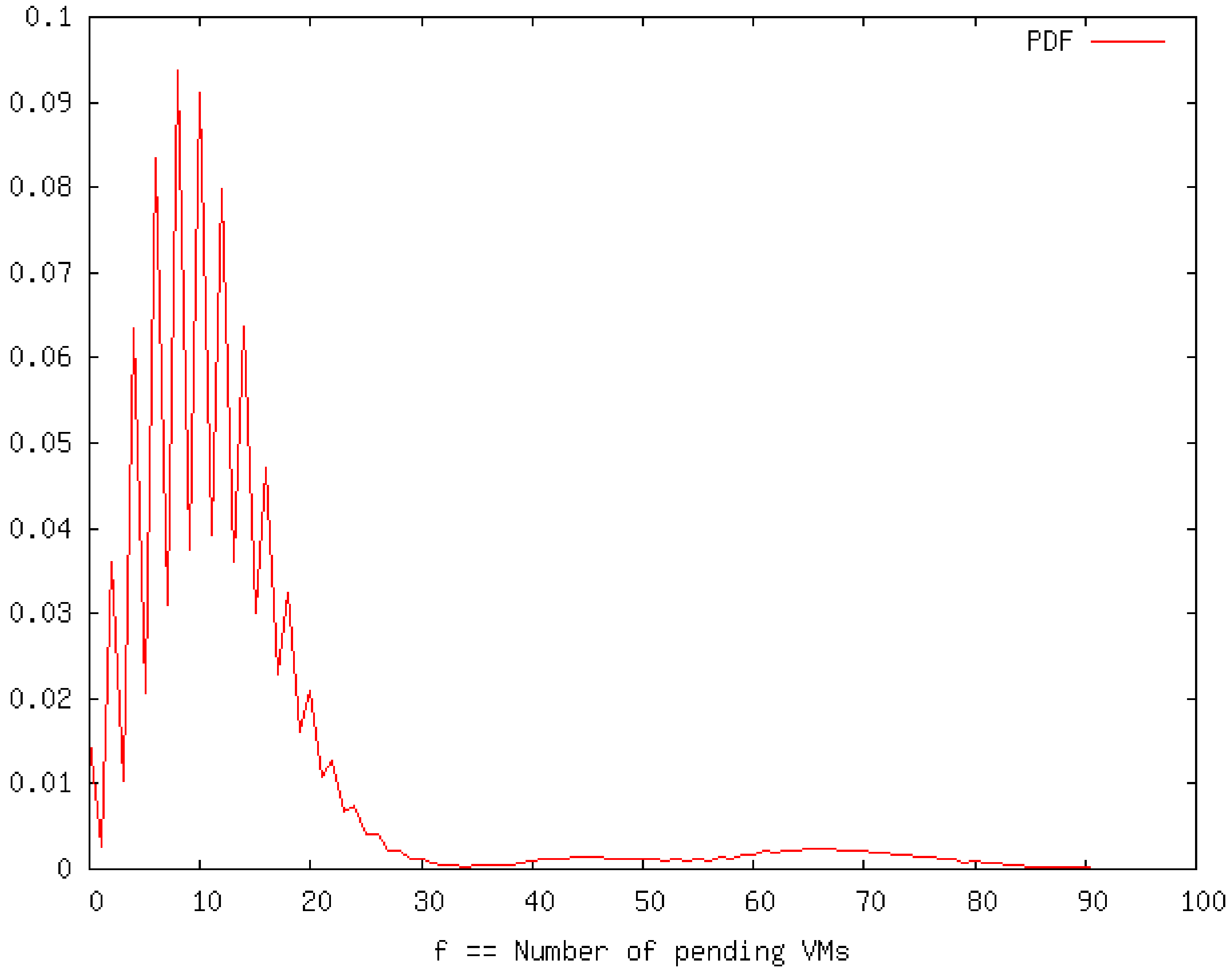
- Assuming that racks fail independently, we can just use convolutions to add P**r**(**f** = **x**) up to P(**f** = **x**)
- For example, P**r**(**f** = [01]) = 0.5 and P**s**(**f** = [02]) = 0.5 can be combined in P(**f** = [0123]) = 0.25

Google

On assumptions and failure correlations

- Notice that we can use the same “trick” used for “sub-racks” to take into account any other failure correlation that we discover is important ...
- ... as well as to use different failure probabilities for different machines, racks, or whatever correlates failures

Example: failure probability of a given job



How come???

Machines	744
… with 1 task	32
… with 2 tasks	423
... with 3 tasks	48
... with 4 tasks	241
... with 5+ tasks	0

Google

Bagging heuristic

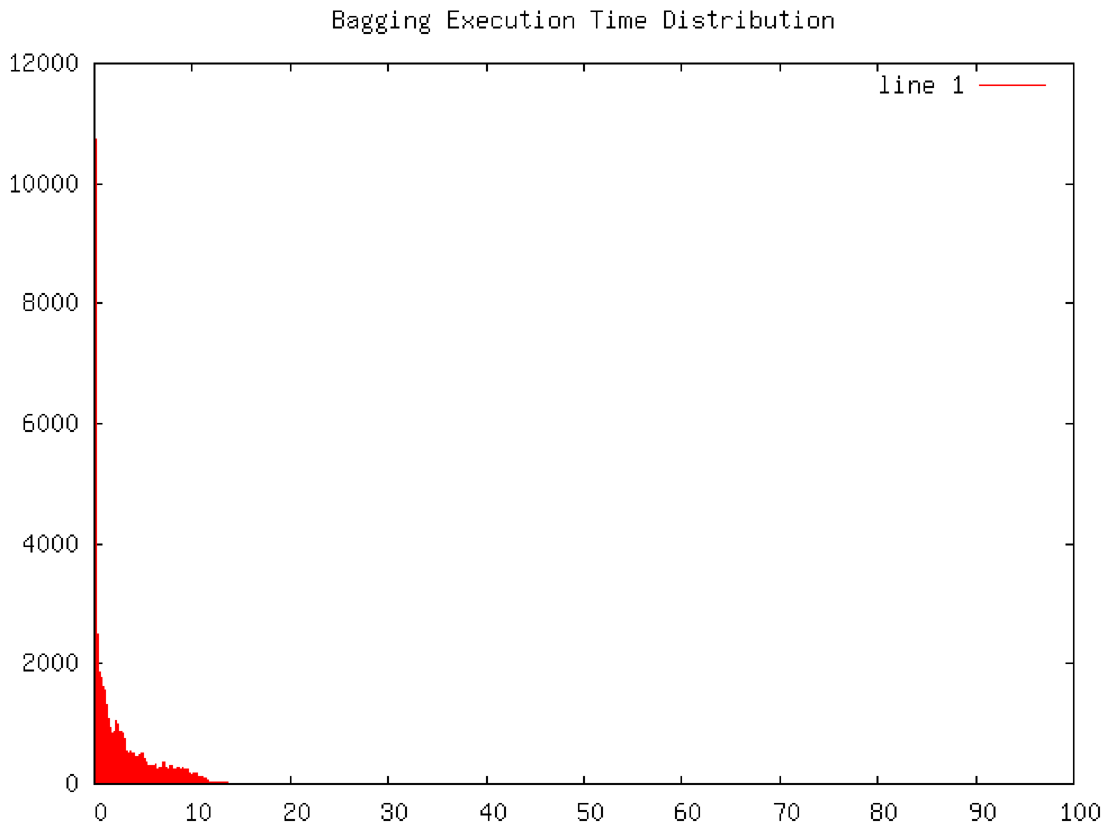
- submission(**S**):
  for each bag **B**:
  if it is possible to find enough backups for **S** + **B**:
  cost[**B**] = BackupCost(**S** + **B**) - BackupCost(**B**)
  cost[ALONE] = BackupCost(**S**)
  select the choice that gives the smallest cost and implement it
- Cost captures how many resources are used
- Our experiments showed improvement if we add a penalty for being creating a new bag
  - cost[ALONE] = K * BackupCost(**S**)

Selecting a backup task

- The selection of backup task is risk-aware
  - We first place a backup task in a new rack (to the bag)
  - If there is no new rack, we place it on a new machine
  - If there is no new machine, we then collocate it with a sibling
- We try to conserve “resource chunks” when we allocate backup tasks
  - That is, we use best fit scheduling for placing backup tasks

Bagging performance

- In the bagging of 78,478 submissions
- Mean = 2.90s, median = 1.76s, max = 92.6s



Resource consumed by the backup tasks

- The aggregated memory reservation of backup tasks is 3.16% of the memory of real tasks

Conclusions

- Greater utilization creates descheduling risk
- We can manage the descheduling risk by strategically placing backup tasks
- This allows us to focus on utilization improvement measures without worrying about descheduling risk

Thanks!!!

JSSPP'2012

- In conjunction with IPDPS
- Shanghai, China, May 25, 2012
- Deadline: February 17, 2012