Weighted Flowtime on Capacitated Machines

Kyle Fox

University of Illinois at Urbana-Champaign

Madhukar Korupolu

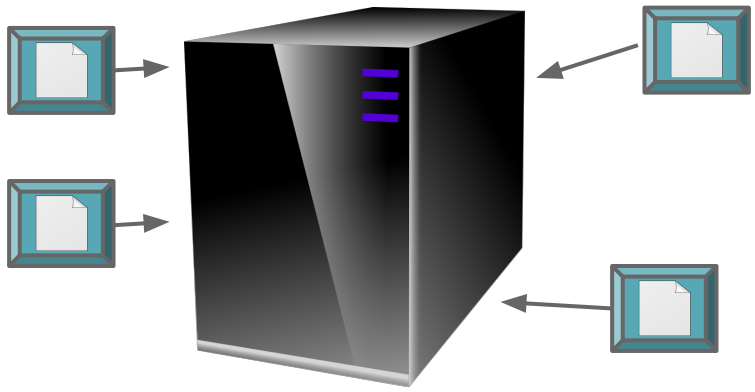
Google Research

Modern Day Machines

- Machines are becoming more powerful and are gaining more resources such as CPU, RAM, etc.
- Machine resources are growing faster than individual job shards

Utilization Gains

- Running one job per machine wastes resources
- Virtualizaton (e.g. VMware) enables multiple jobs to run on the same machine
- Helps consolidate hardware, reduces space, reduces cooling costs, *etc*.



An Online Cloud Scenario

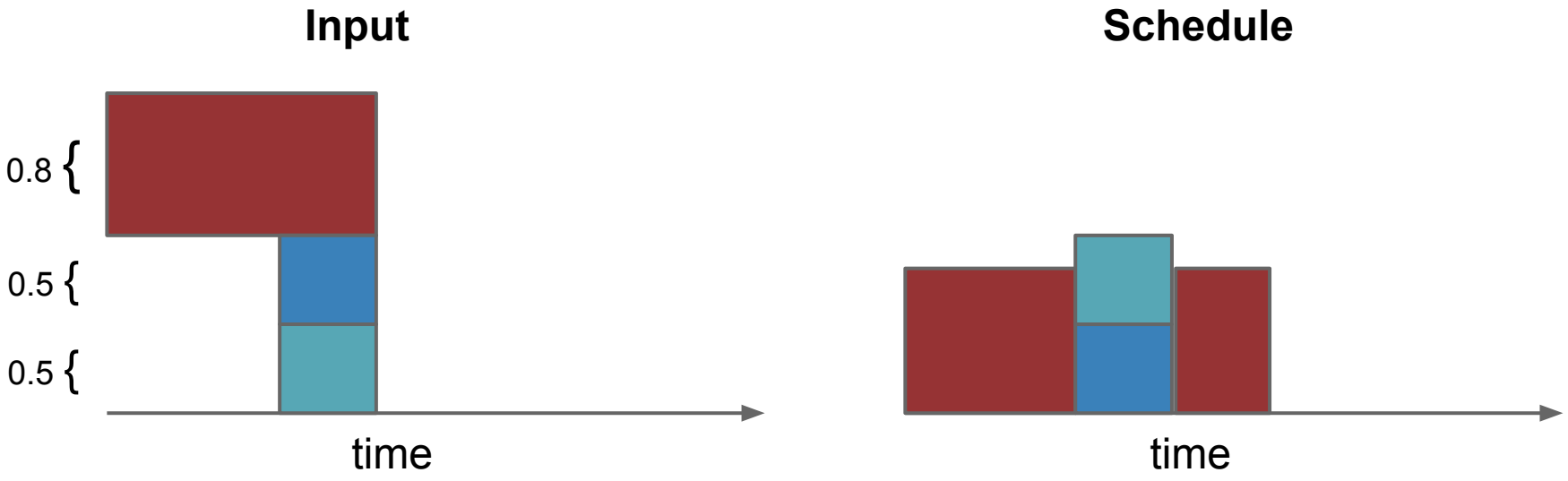
- Jobs of different durations and resource requirements arrive over time
- Need to schedule them on a large collection of machines
- Want to measure quality of the schedule (total flowtime)

Capacitate Flowtime Problem

- Jobs 1,…,n
- Job j arrives at time r_j and needs scheduling for p_j time units
- Can preempt and resume jobs

Capacitated Flow Time Problem

- Server is given capacity 1
- Job j has height h_j
- Can schedule an arbitrary number of jobs *simultaneously* assuming heights add up to at most 1



Capacitated Flow Time Problem

- Common objective: weighted flow time
- Job j gets weight w_j
- C_j denotes completion time of job j
- Total weighted flow time: $\sum_j w_j (C_j - r_j)$

Models and Objectives

- Online scheduling: scheduler only learns p_j, w_j, and h_j when j arrives
- Want to minimize competitive ratio of weighted flow time
- Model can be extended to m (identical) machines
- Prior work assumes one job per machine at a time

Prior Work: Single-job-at-a-time Model

Capacitated	Weights	Machines	Competitive Ratio
no	unit	single	1 [forklore, (SRPT)]
no	unit	multiple	$\Omega(\log n/m + \log P)$, $O(\log \min\{n/m, P\})$ [Leonardi, Raz '07]
no	arbitrary	single	$\omega(1)$ [Bansal, Chan '09] $O(\log^2 P)$ [Chekuri, Khanna, Zhu '01] $O(\log W)$, $O(\log n + \log P)$ [Bansal, Dhamdhere '07]
no	arbitrary	multiple	$\Omega(\min\{W^{1/2}, P^{1/2}, (n/m)^{1/4}\})$ [CKZ '01]

P - ratio of max to min processing time
W - number of distinct weights

Resource Augmentation

- Gives online algorithms a fighting chance on worst case instances

- *s*-speed *c*-competitive: for any input *I*, our algorithm's weighted flow time running at speed *s* is at most *c* times the optimal running at unit speed

Prior Work: Single-job-at-a-time Model

Capacitated	Weights	Machines	Competitive Ratio
no	unit	single	1 [forklore, (SRPT)]
no	unit	multiple	$\Omega(\log n/m + \log P)$, $O(\log \min\{n/m, P\})$ [Leonardi, Raz '07] **(1+ε)-speed O(1/ε)-competitive [Chekuri, *et al.* '04]**
no	arbitrary	single	$\omega(1)$ [Bansal, Chan '09] $O(\log^2 P)$ [Chekuri, Khanna, Zhu '01] $O(\log W)$, $O(\log n + \log P)$ [Bansal, Dhamdhere '07] **(1+ε)-speed O(1/ε)-competitive [Becchetti, *et al.* '01]**
no	arbitrary	multiple	$\Omega(\min\{W^{1/2}, P^{1/2}, (n/m)^{1/4}\})$ [CKZ '01] **(1+ε)-speed $O(1/\varepsilon^2)$-competitive [Bussema, Torng '06]**

First Result

- Online capacitated flow time is hard!
- Theorem: Any randomized online algorithm for the CFT problem has competitive ratio $\Omega(P)$ even when all jobs have unit weight.
- In contrast to single-job-at-a-time model where SRPT is optimal

Need Resource Augmentation

- *u*-capacity *c*-competitive: for any input *I*, our algorithm's weighted flow time running with capacity *u* is at most *c* times the optimal running at unit speed

- *s*-speed *u*-capacity *c*-competitive if *c*-competitive with both *s*-speed and *u*-capacity

More Lower Bound Results

Capacity	Weights	Randomed / Deterministic	Competitive Ratio
1.2 - ε	unit	randomized	Ω(P)
1.5 - ε	unit	randomized	Ω(log n + log P)
2 - ε	**arbitrary**	**deterministic**	**ω(1)**

CFT Problem: Upper Bound Algorithms and Results

- Highest Residual Density First - a $(2+\varepsilon)$-capacity $O(1/\varepsilon)$-competitive simple greedy online algorithm

- High Priority Category Scheduling - a 1.75-speed $O(1)$-competitive online algorithm

- Knapsack Category Scheduling - a $(1+\varepsilon)$-speed $(1+\varepsilon)$-capacity $O(1/\varepsilon^3)$-competitive online algorithm

CFT Problem: Remarks on New Algorithms

- Algorithms use a novel combination of
  - Ordering jobs by density
  - Grouping jobs with similar heights
  - Knapsack algorithms and analyses
- All algorithms work with weighted jobs
- Results extend to multiple unrelated machines
  - Actual difficulty found in single machine case

Alg1: Highest Residual Density First (HRDF)

- Pack jobs in decreasing order of density (weight / height * remaining-proc.-time)
- Generalizes SRPT

HRDF:

/* $Q^A(t)$: Jobs alive at time t for HRDF */

/* $p_j^A(t)$: Remaining processing for j at time t */

1. /* Assign residual densities */
   For each $j \in Q^A(t)$: $d_j^A(t) \leftarrow w_j / (h_j\, p_j^A(t))$

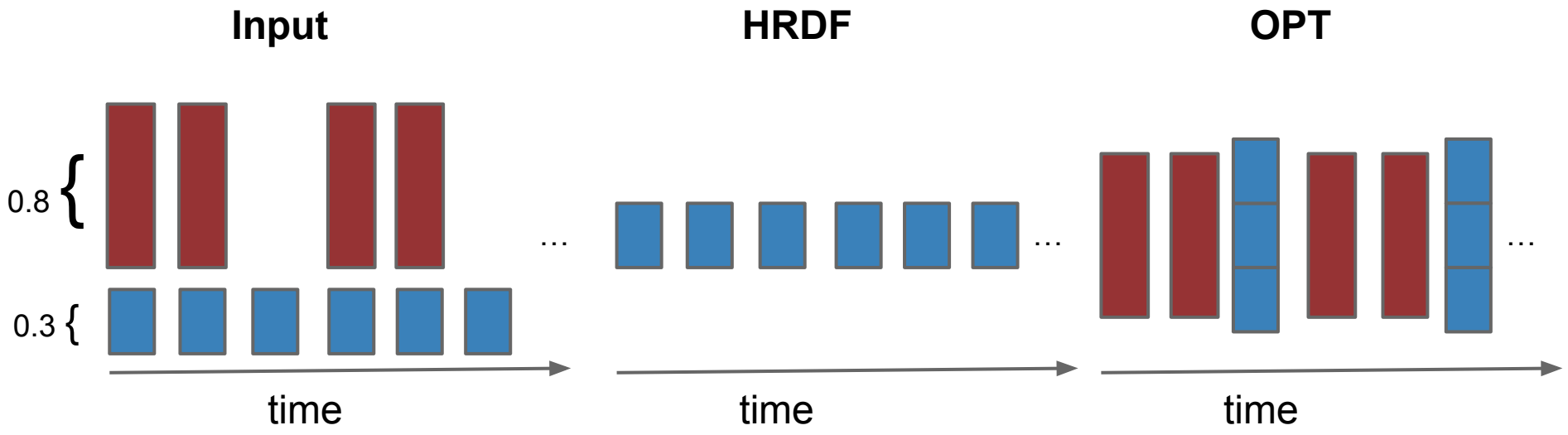
2. Sort $Q^A(t)$ by $d_j^A(t)$ descending and pack jobs from $Q^A(t)$ up to capacity

Analysis of HRDF

- Theorem: HRDF is $(2+\varepsilon)$-capacity $O(1/\varepsilon)$-competitive
- Proof uses a potential function argument
  - Run HRDF with augmentation and optimal schedule simultaneously
  - Bound increases to our algorithm's weighted flow time and the potential function
- Need to show HRDF fills machine as effectively as the optimal schedule
- Can we do better?

HRDF Needs Extra Capacity

- Can we compete using only speed?
- Greedily packing by density can cause jobs to pile up, even with small augmentations in speed
- Machine resources are being wasted



Alg2: High Priority Category Scheduling (HPCS)

- κ : number of categories as determined by analysis
- For talk, ignore jobs with height $\leq 1/\kappa$

HPCS:

1. /* Divide jobs into categories by height */
   For each $1 \leq i < \kappa$:
   $Q_i^A(t) \leftarrow \{j \in Q^A(t) : 1/(i+1) < h_j \leq 1/i\}$
   For each $j \in Q_i^A(t) : d_j^A(t) \leftarrow i * w_j / p_j^A(t)$

2. Pack jobs from highest weight category $Q_i^A(t)$ in descending order of residual density

Analysis of HPCS

- Theorem: HPCS is 1.75-speed O(1)-competitive
- New definition of density gives each category *i* job an effective height of 1/*i* in the potential function analysis
- Need to show HPCS uses more effective height than optimal schedule
- Model effective resources used by optimal schedule as a knapsack instance
  - Easy to show 1.73 upper bound

Alg3: Knapsack Category Scheduling (KCS)

- Effective resource use for each job is too high compared to real resource use
- Idea: place jobs in tight categories using geometrically decreasing height
- Jobs with height ≤ $\varepsilon/2$ have their own category (which we ignore for this talk)
- Solve a knapsack instance where each category *i* job *j* has value equal to total weight of less dense category *i* jobs

Alg3: Knapsack Category Scheduling (KCS)

KCS:

1. For each integer $i \geq 0$ with $(1+\varepsilon/2)^i < 2/\varepsilon$:

   $Q_i^A(t) \leftarrow \{j \in Q^A(t) : 1/(1+\varepsilon/2)^{i+1} < h_j \leq 1/(1+\varepsilon/2)^i\}$

   For each $j \in Q_i^A(t) : d_j^A(t) \leftarrow (1+\varepsilon/2)^i * w_j / p_j^A(t)$

2. Initialize knapsack $K(t)$ with capacity $(1+\varepsilon)$

   For each category i; for each $j \in Q_i^A(t)$

   $Q_i^{Aj}(t) \leftarrow \{k \in Q_i^A(t) : d_k^A(t) > d_j^A(t)\}$

   Add item j to $K(t)$ with size h_j and value

   $$w_j + 1/(1+\varepsilon/2)^i \sum_{k \in (QiA(t) \setminus QiAj(t) \setminus \{j\})} w_k$$

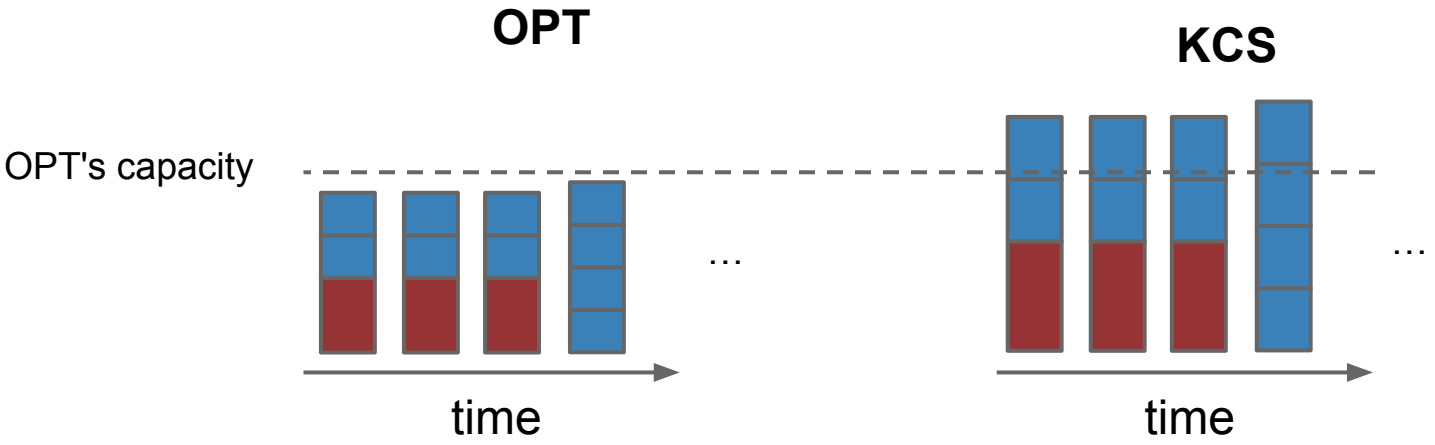
3. Pack jobs from a $(1-\varepsilon/2)$-approximation for $K(t)$

Alg3: Knapsack Category Scheduling (KCS)

- Knapsack value of item *j* intuitively represents weight of jobs that cannot be scheduled until *j* is complete
- Pragmatically, values are chosen to maximize the decrease of the potential function

Analysis of KCS

- Theorem: KCS is $(1+\varepsilon)$-speed $(1+\varepsilon)$-capacity $O(1/\varepsilon^3)$-competitive
- KCS can pick from same categories as OPT
- When optimal schedule picks T_i jobs with total height H_i from category i, KCS can pick T_i most dense jobs using height $(1+\varepsilon/2)H_i$



Analysis of KCS

- KCS has $(1+\varepsilon)$-speed and picks an approximately best set of jobs
- Now category i jobs have effective height $1/(1+\varepsilon/2)^i$
- Total effective height scheduled by KCS is greater than for the optimal algorithm
- KCS prioritizes higher density jobs within each category

Summary of Results

Algorithm or Lower Bound	Capacity	Speed	Competitive Ratio
lower bound	$1.2 - \varepsilon$	1	$\Omega(P)$
lower bound	$1.5 - \varepsilon$	1	$\Omega(\log n + \log P)$
lower bound	$2 - \varepsilon$	1	$\omega(1)$
Highest Residual Density First	$2 + \varepsilon$	1	$O(1/\varepsilon)$
High Priority Category Scheduling	1	1.75	$O(1/\varepsilon)$
Knapsack Category Scheduling	**$1 + \varepsilon$**	**$1 + \varepsilon$**	**$O(1/\varepsilon^3)$**

Open Problems

- Can we do: $(1+\varepsilon)$-speed $f(\varepsilon)$-competitive online algorithm?

- Multi-dimensional heights
  - Heights are d-dimensional vectors
  - Can simultaneously schedule jobs given no single dimension of their vector sum surpasses the server's capacity

Thank you