On the Latency-Accuracy Tradeoff in Approximate MapReduce Jobs

Juan F. Pérez
Universidad del Rosario, Colombia

Robert Birke and **Lydia Chen**
IBM Research Zurich, Switzerland

IEEE INFOCOM – May, 2017

Outline

What?

MapReduce Jobs

- Large data sets
- Exploit large number of parallel resources
- Open source implementations (Hadoop)
- Well studied for batch jobs
- **Online** job execution
- Deadlines: constraints on response time

MapReduce Jobs

- Large data sets
- Exploit large number of parallel resources

- **Online** job execution
- Deadlines: constraints on response time

Online MapReduce Jobs

- Deadlines: constraints on response time
- Process **all** data $\rightarrow$ *exact* answer
- **Approximate** Computing
- Process **some** data $\rightarrow$ *approximate* answer
- Save computing time and resources
- Comply with deadlines

Why?

Latency-Accuracy Tradeoff

- Drop more data $\rightarrow$ Reduce latency $\rightarrow$ Reduce accuracy
- Drop less data $\rightarrow$ Increase latency $\rightarrow$ Increase accuracy
- Solve the **Latency-Accuracy Tradeoff**

Goal?

Solve Latency-Accuracy Tradeoff

- Given a *Latency* objective → Determine *Achievable Accuracy*
- Given an Accuracy objective → Determine Achievable Latency
- Additional considerations:
    - Jobs arrive in a random fashion
    - Queueing times waiting for jobs in front to be processed
    - How to drop data for approximate computations?

Solve Latency-Accuracy Tradeoff

- Given a *Latency* objective $\rightarrow$ Determine *Achievable Accuracy*
- Given an Accuracy objective $\rightarrow$ Determine Achievable Latency
- Additional considerations:

  Jobs arrive in a random fashion

  Queueing times waiting for jobs in front to be processed

  How to drop data for approximate computations?

How?

Our Approach to the Latency-Accuracy Tradeoff

- Stochastic models to predict **Job Latency**
- Capture Online MapReduce operation
- Capture Approximate Computing
- Data Drop options

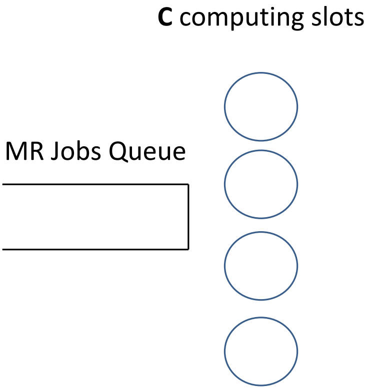
Our Approach to the Latency-Accuracy Tradeoff

- Stochastic models to predict **Job Latency**
- Capture Online MapReduce operation
- Capture Approximate Computing
- Data Drop options

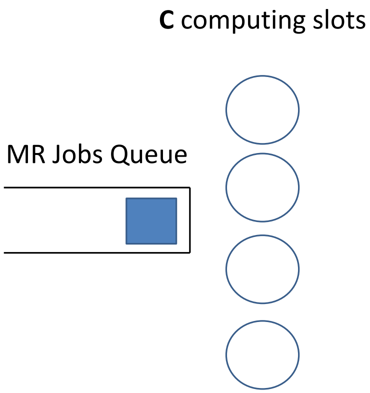
Outline

Basic Operation

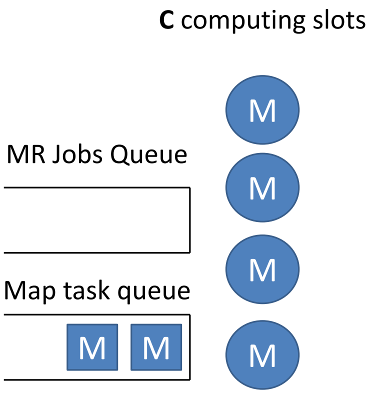
Basic Operation



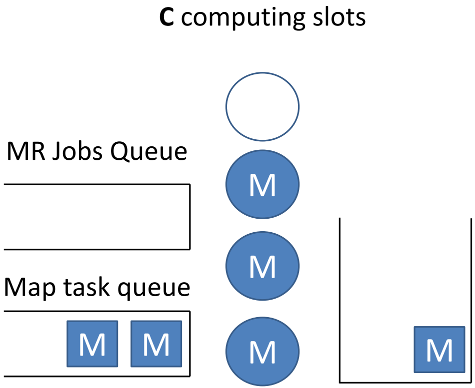
Basic Operation



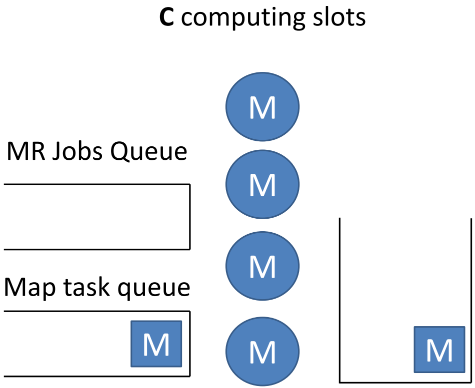
Basic Operation



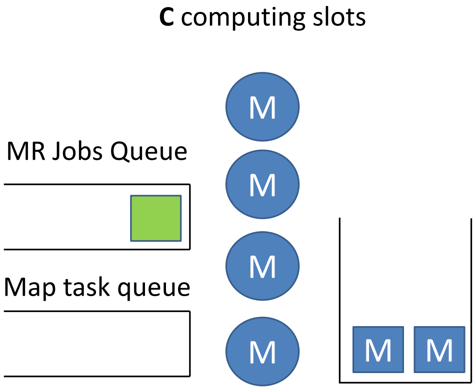
Basic Operation



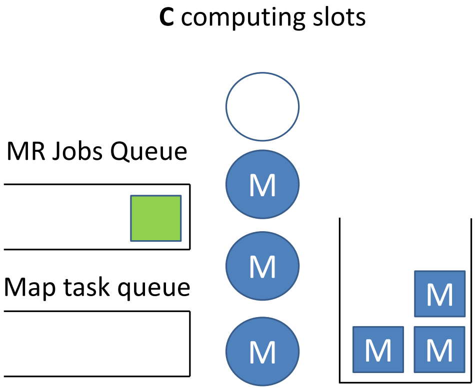
Basic Operation



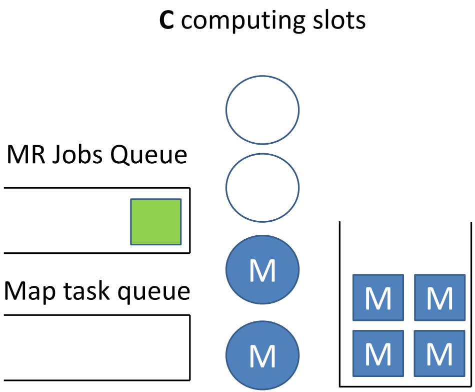
Basic Operation



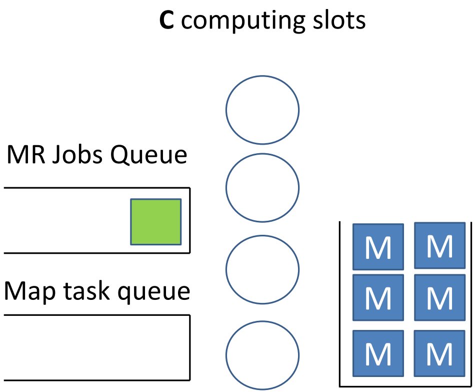
Basic Operation



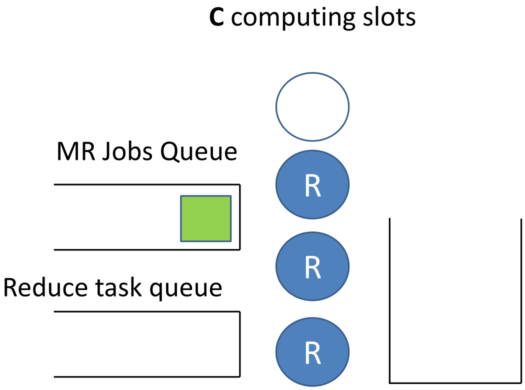
Basic Operation



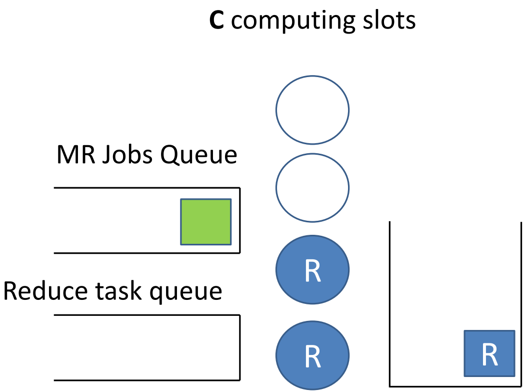
Basic Operation



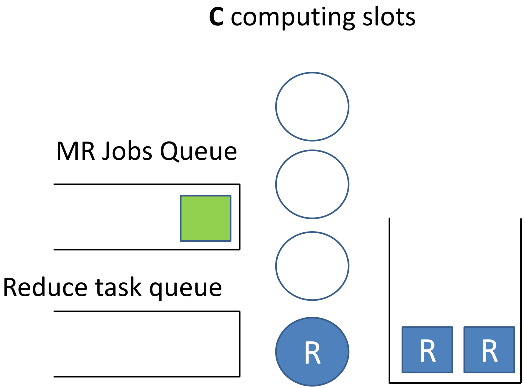
Basic Operation



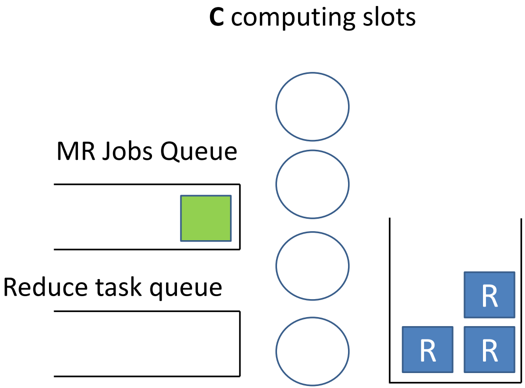
Basic Operation



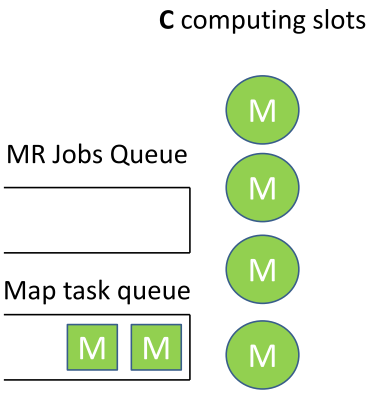
Basic Operation



Basic Operation

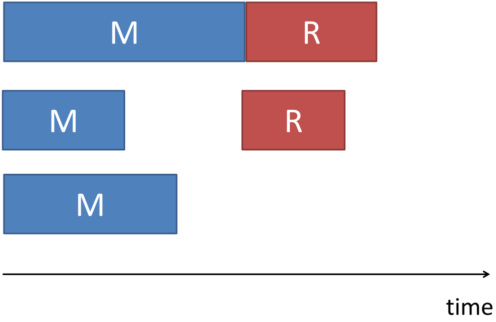


Basic Operation



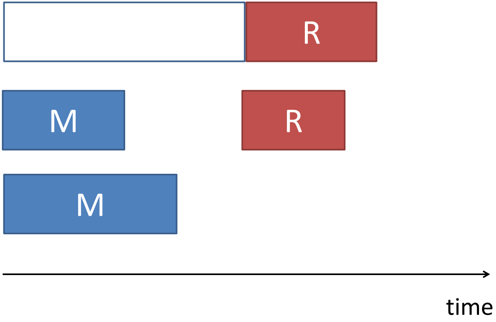
Approximate Computing

Data Drop

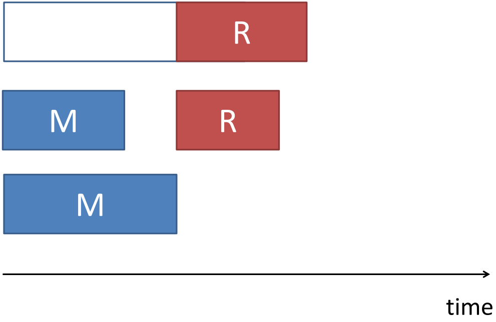


Early Dropping

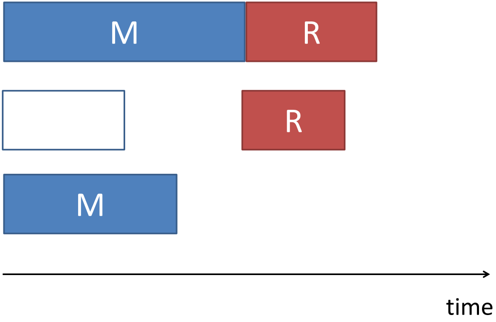
Data Drop: Task drop - Early



Data Drop: Task drop - Early

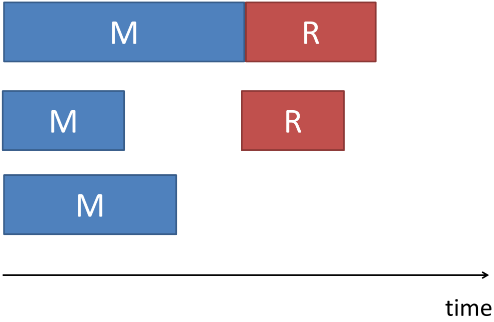


Data Drop: Task drop - Early

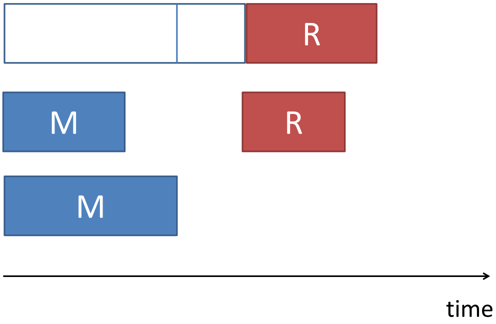


Straggler Dropping

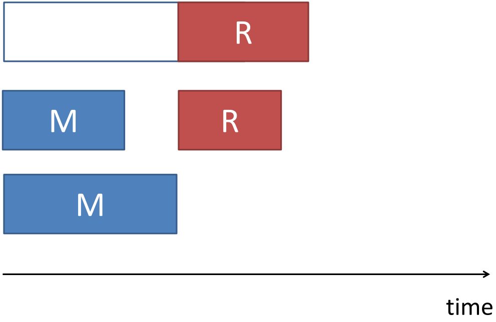
Data Drop: Task drop - Straggler



Data Drop: Task drop - Straggler

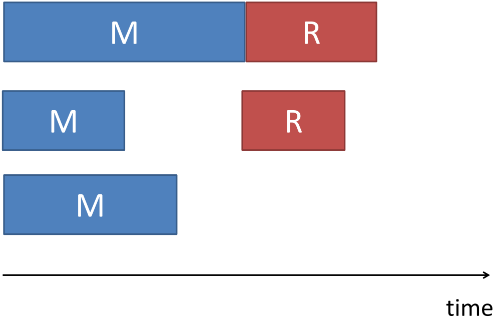


Data Drop: Task drop - Straggler

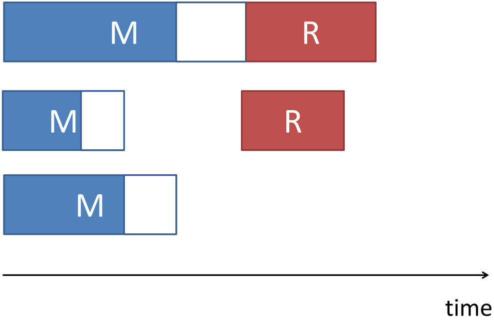


Input Sampling

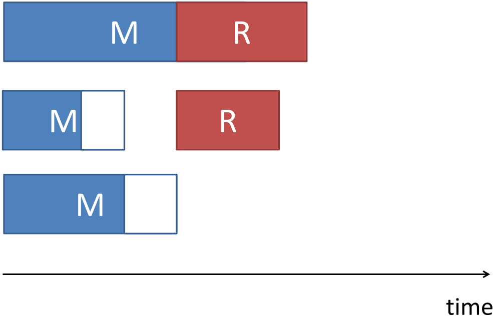
Data Drop: Input Sampling



Data Drop: Input Sampling



Data Drop: Input Sampling



Error

$$t_{\overline{N}_m-1,1-\alpha/2}\sqrt{N_m\left(\left(\frac{1}{\theta_m}-1\right)s_u^2+\left(\frac{1}{\eta_m}-1\right)\overline{s}_i^2\right)},$$

Outline

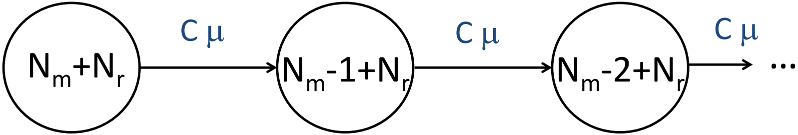
Basic Operation - Main parameters

- C computing slots
- N_m: number of map tasks
- N_r: number of reduce tasks
- $1/\mu_m$: mean map task execution time
- $1/\mu_r$: mean reduce task execution time

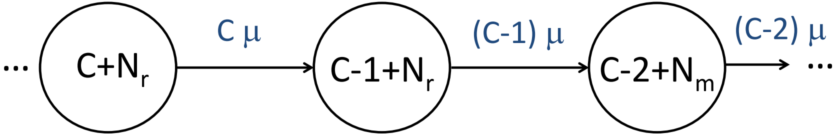
Basic Operation - Job Service Time

- Job Service Time: **Phase-type** distribution
- Phase: number of tasks to complete

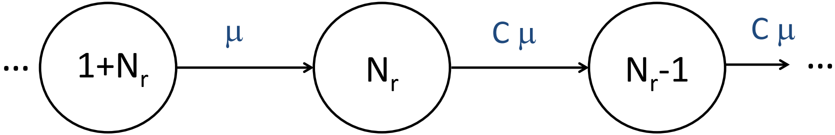
Basic Operation - Job Service Time



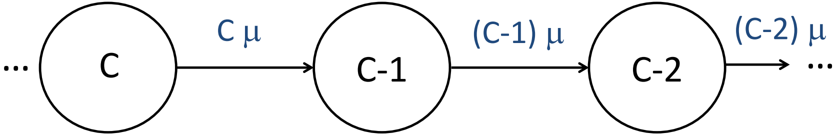
Basic Operation - Job Service Time



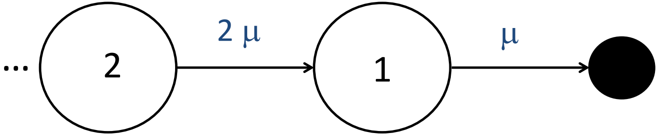
Basic Operation - Job Service Time



Basic Operation - Job Service Time



Basic Operation - Job Service Time



Modeling Approximate Computation

Introducing Early Task Dropping

- Execute fraction θ_m map tasks only
- Drop fraction $1 - \theta_m$ map tasks
- Effective number of map tasks: $\overline{N}_m = \lceil N_m \theta_m \rceil$
- Replace N_m by $\overline{N}_m$

Introducing Input Sampling

- Process fraction η_m of each map task data only
- Ignore fraction $1 - \eta_m$ of each map task data
- Effective mean map task execution time: $\frac{1}{\overline{\mu}_m} = \frac{\eta_m}{\mu_m}$
- Replace μ_m by $\overline{\mu}_m$

Early Dropping + Input Sampling

Transition Rates:

$$f(n) = \begin{cases} C\overline{\mu}_m, & \overline{N}_r + C < n \leq \overline{N}_r + \overline{N}_m, \\ (n - \overline{N}_r)\overline{\mu}_m, & \overline{N}_r < n \leq \overline{N}_r + C, \\ C\overline{\mu}_r, & C < n \leq \overline{N}_r, \\ n\overline{\mu}_r, & 0 < n \leq C. \end{cases}$$

Straggler Dropping

- Start with $N_m + N_r$ tasks
- Decrease one by one until $N_m - (\overline{N}_m - 1) + N_r$ are left
- Jump down to N_r
- Effective number of map tasks executed: $\overline{N}_m$

Straggler Dropping + Input Sampling

Transition Rates:

$$f(n) = \begin{cases} \min\{n - N_r, C\}\overline{\mu}_m, & N_r + N_m - \overline{N}_m + 1 \leq n \\ & \qquad \leq N_r + N_m, \\ \min\{n, C\}\overline{\mu}_r, & N_r - \overline{N}_r + 1 \leq n \leq N_r. \end{cases}$$

Analysis

Analysis

- FCFS scheduling
- Job service time: phase-type (PH) - $(\alpha, \boldsymbol{G})$
- Job arrival process: Markovian arrival process (MAP)
- MAP/PH/1 queue
- Sengupta's paper: age-based analysis
- $(X(t), A(t), S(t))$: age of job in service, arrival process phase, service phase

Analysis

Solve a matrix-integral equation

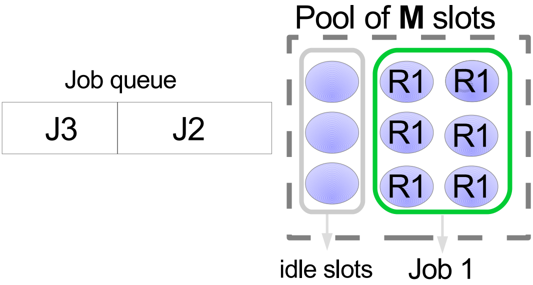
$$\boldsymbol{T} = \boldsymbol{I}_d \otimes \boldsymbol{G} + \int_0^\infty \exp(\boldsymbol{T}x) \left(\exp(\boldsymbol{D}_0 x) \boldsymbol{D}_1 \otimes \tilde{\boldsymbol{G}} \right) dx,$$

- Obtain waiting time distribution $(\boldsymbol{\beta}_w, \boldsymbol{B}_w)$
- Known service-time distribution $(\boldsymbol{\alpha}, \boldsymbol{G})$
- Obtain response time distribution $(\boldsymbol{\beta}_r, \boldsymbol{B}_r)$:

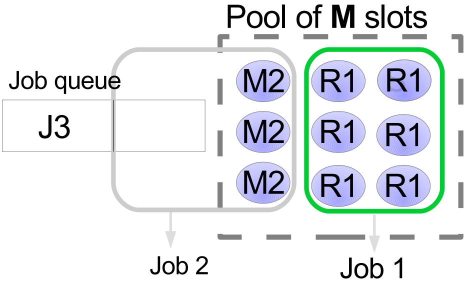
$$\boldsymbol{\beta}_r = [\phi \boldsymbol{\beta}_w \quad (1-\phi)\boldsymbol{\alpha}], \quad \boldsymbol{B}_r = \begin{bmatrix} \boldsymbol{B}_w & (-\boldsymbol{B}_w \mathbf{1})\boldsymbol{\alpha} \\ \mathbf{0} & \boldsymbol{G} \end{bmatrix},$$

Overlapping Execution

FCFS Scheduling - Non-overlapping



Overlapping scheduling



- Allow jobs to start Map phase as soon as slots become available.
- At most 2 jobs in service: one in Map phase, one in Reduce phase.

Overlapping Scheduling

- No longer a MAP/PH/1 queue
- Service times depend on system state
- High load: high chance back-to-back jobs
- Low load: similar to non-overlapping

Overlapping Scheduling

- Generalize Sengupta's analysis
- $(X(t), A(t), M(t), R(t), Y(t))$
- $M(t)$: number of remaining map tasks of oldest job
- $R(t)$: number of remaining reduce tasks of oldest job
- $Y(t)$: number of remaining map tasks of youngest job

Overlapping Scheduling

Three phases:

- M one job, map phase
- R one job, reduce phase
- R/M two jobs, oldest in reduce phase, youngest in map phase

Overlapping Scheduling

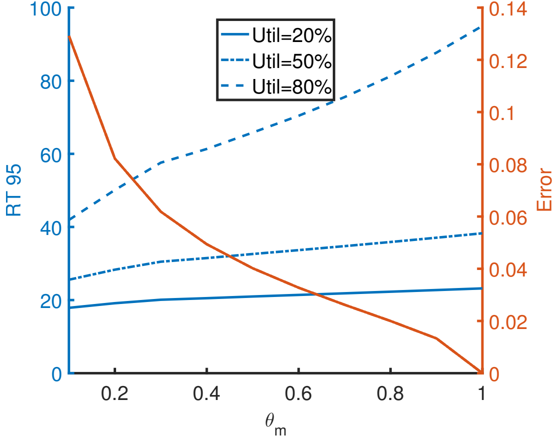
(Steady-state) Service time distribution: PH($\boldsymbol{\beta}_s$, $\boldsymbol{B}_s$)

$$\boldsymbol{B}_s = \begin{bmatrix} \boldsymbol{Q}^{R/M,R/M} & \boldsymbol{Q}^{R/M,M} & \boldsymbol{Q}^{R/M,R^*} \\ \mathbf{0} & \boldsymbol{Q}^{M,M} & \boldsymbol{Q}^{M,R^*} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{B}^{R^*,R^*} \end{bmatrix}, \tag{1}$$

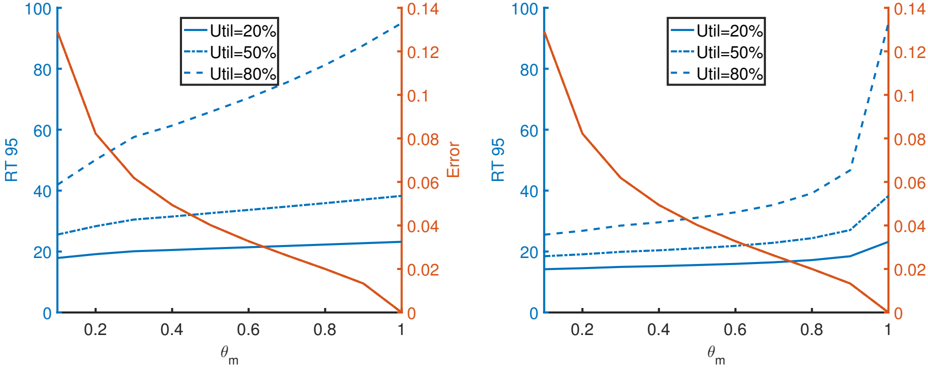
$$\boldsymbol{\beta}_s = \begin{bmatrix} \boldsymbol{\beta}_s^{R/M} & \boldsymbol{\beta}_s^{M} & \boldsymbol{\beta}_s^{R^*} \end{bmatrix}. \tag{2}$$

Outline

FCFS - Early Dropping

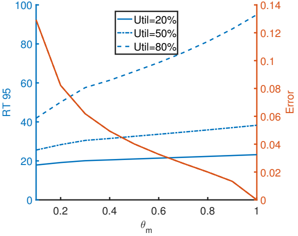


FCFS - Early vs Straggler Dropping

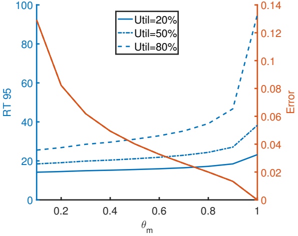


(a) Early dropping (b) Straggler dropping

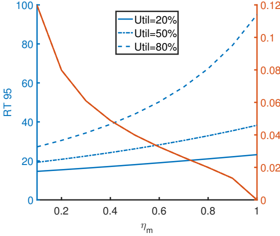
FCFS - Early vs Straggler Dropping vs Input Sampling



(a) Early dropping



(b) Straggler dropping



(c) Input sampling

FCFS - Early vs Straggler Dropping vs Input Sampling

Observation 1

Best Mechanisms are:

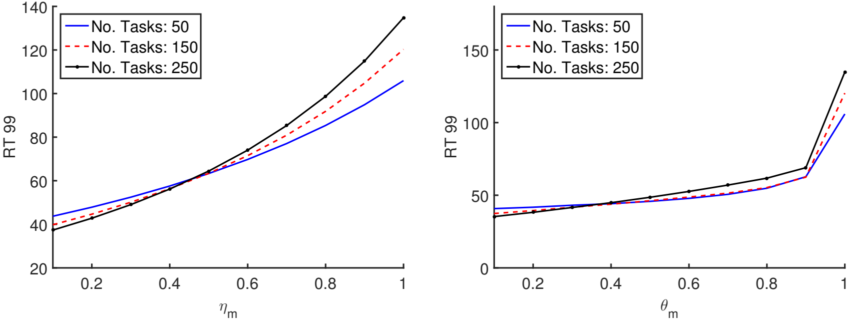
1. Straggler dropping
2. Input sampling
3. Early dropping

Observation 2

An $x\%$ data/task dropping removes

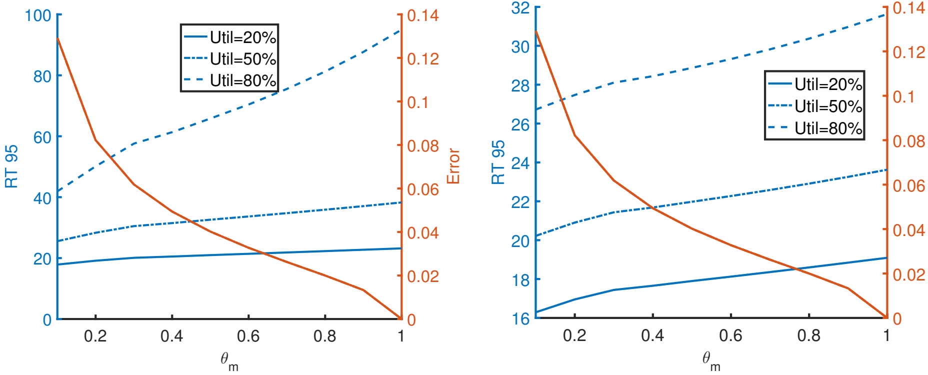
1. $x\%$ **longest** tasks (straggler dropping)
2. $x\%$ **all** tasks, including longest (input sampling)
3. $x\%$ tasks (early dropping)

Varying number of tasks



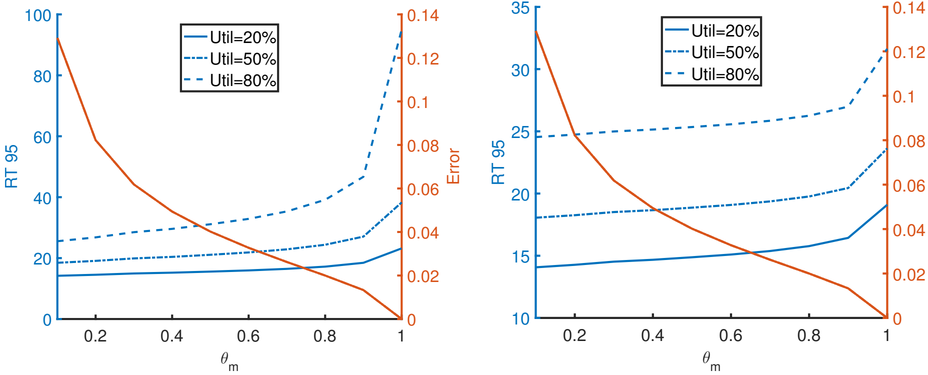
(a) Input sampling (b) Straggler dropping

FCFS vs. Overlapping (Early dropping)



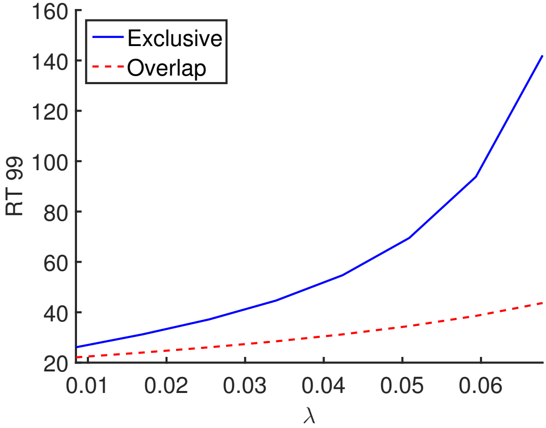
(a) FCFS (b) Overlapping

FCFS vs. Overlapping (Straggler dropping)



(a) FCFS (b) Overlapping

FCFS vs. Overlapping (No approximation)



Outline

Wrap-up

Model captures

- Straggler Dropping > Input Sampling > Early Dropping
- Overlapping > FCFS Scheduling
- Latency - Accuracy Tradeoff

Given a:

- **Latency Budget**: what accuracy can be expected/offered?

Wrap-up

Model captures

- Straggler Dropping > Input Sampling > Early Dropping
- Overlapping > FCFS Scheduling
- Latency - Accuracy Tradeoff

Given a:

- **Latency Budget**: what accuracy can be expected/offered?
- **Accuracy Budget**: what latency can be expected/offered?

Wrap-up

Model captures

- Straggler Dropping > Input Sampling > Early Dropping
- Overlapping > FCFS Scheduling
- Latency - Accuracy Tradeoff

Given a:

- **Latency Budget**: what accuracy can be expected/offered?
- **Accuracy Budget**: what latency can be expected/offered?

Wrap-up

Model captures

- Straggler Dropping > Input Sampling > Early Dropping
- Overlapping > FCFS Scheduling
- Latency - Accuracy Tradeoff

Given a:

- **Latency Budget**: what accuracy can be expected/offered?
- **Accuracy Budget**: what latency can be expected/offered?

Thank you!