# Wide-area Analytics with Multiple Resources

**Chien-Chun Hung**, Ganesh Ananthanarayanan,
Leana Golubchik, Minlan Yu, Mingyang Zhang
*USC, Microsoft Research, Harvard*

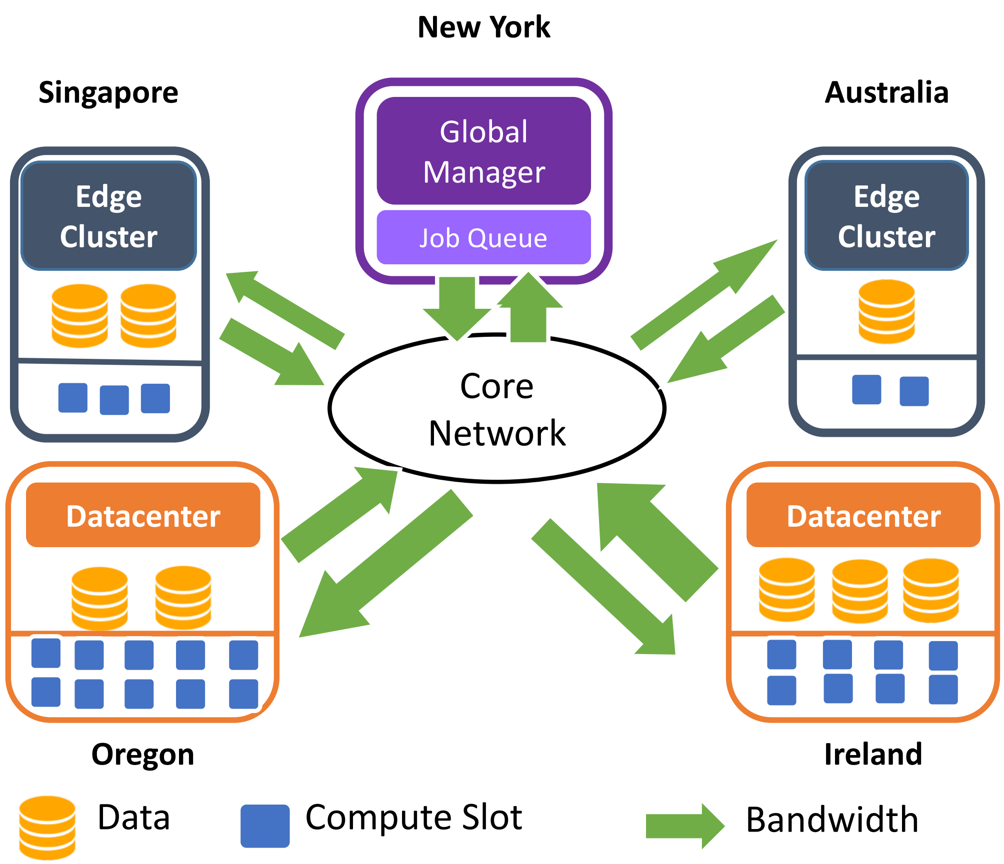
*April 24$^{th}$, 2018*

# Wide-area Data Analytics Overview

- User session logs analysis
- System health monitoring, troubleshooting

- Application data: *generated*, *stored* and *processed* across multiple locations
- ***Fast response time*** of wide-area data analytics is critical for applications!

# Wide-Area Data Analytics Architecture




***Heterogeneity***

- Num. of slots differ by up to ***2*** order of magnitude
- Network bandwidths differ by up to ***18X – 25X***
- Data volumes differ by up to ***22X***

***Schedule Jobs with data-parallel tasks (map, reduce)***

- **Task placement**
- **Job scheduling**

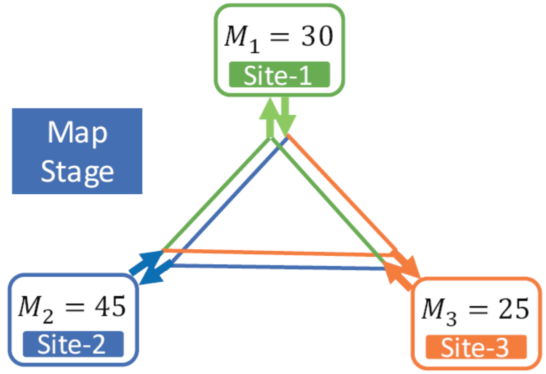
# Existing Solutions and Limitations

- ***Centralized*** approach
  - Aggregate all required data at ***a single site*** up front
  - Incur lots of data transfer and significant delay
- ***In-Place*** approach
  - Move computation to meet data locality
  - Perform poor under data-resource ***mismatch***
- ***Network-centric*** approach [Iridium-sigcomm15]
  - Distribute tasks to ***minimize network transfer delay***
  - Ignore computation capacity constraint

# Challenge 1:
# Heterogeneity in Resource Distribution, and Mismatch with Resource Demands

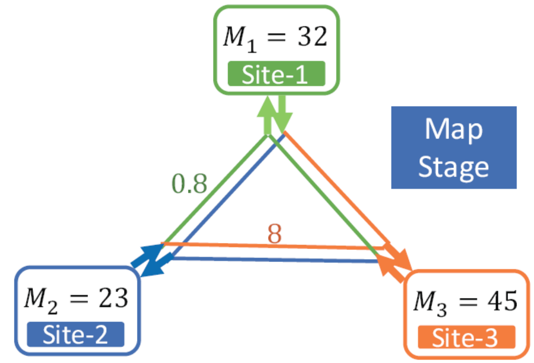
# Map-task Placement Example

| | Site-1 | Site-2 | Site-3 |
|---|---|---|---|
| #Slots | 5 | 1 | 2 |
| Up BW | 10 | 10 | 10 |
| Down BW | 1 | 10 | 10 |
| Input volume | 12 | 18 | 10 |
| #Map tasks | 30 | 45 | 25 |




**Network-Centric Approach**

Places tasks locally for data




**Optimal Approach**

Shifts compute loads

| | Network-Centric | | | Optimal | | |
|---|---|---|---|---|---|---|
| | Site 1 | Site 2 | Site 3 | Site 1 | Site 2 | Site 3 |
| Netw. duration | 0 | 0 | 0 | 0.8 | **0.88** | 0.8 |
| Comp. duration | 6 | **45** | 13 | 7 | **23** | **23** |
| Total duration | **45** | | | **23.88** | | |

**Optimize both network and computation time!**

# Task Assignment Solution

- Break-down: ***network transfer*** followed by ***computation***
- Network transfer time
  - Transfer time = data size / network bandwidth
  - $N^2$ data upload and download transfer given ***N*** sites
  - Focus on minimizing the the bottleneck of all transfer
- Computation time
  - Estimated based on **#waves**, i.e., ***#tasks / #slots***
  - Focus on minimizing the bottleneck of computation across sites

- Formulate task placement as an **Linear Program (LP)** to minimize ***network transfer time + computation time***

# Map-task Placement LP

| | |
|---|---|
| $\boldsymbol{m_{x,y}}$ | Fraction of map-tasks placed at site y that read data from site x |
| $T_{aggr}$ | Network duration for input data transfer |
| $T_{map}$ | Computation duration for map-stage |
| $S_x, B_x{}^{up}, B_x{}^{down}, I_x{}^{input}$ | #slots, up/down b/w, data volume at site x; $I^{input} = \sum_x I_x{}^{input}$ |
| $n_{map}; t_{map}$ | #map-tasks; duration of a map-task |

$$\min_{m_{x,y}} T_{aggr} + T_{map}$$

**Minimize total duration (net. + comp.)**

$$s.t.$$

$$T_{aggr} \geq \frac{I^{input} \times (\sum_{y \neq x} m_{x,y})}{B_x{}^{up}}, \forall x$$

Upload transfer duration

$$T_{aggr} \geq \frac{I^{input} \times (\sum_{y \neq x} m_{y,x})}{B_x{}^{down}}, \forall x$$

Download transfer duration

$$T_{map} \geq t_{map} \times \left(\frac{n_{map} \times \sum_y m_{y,x}}{S_x}\right), \forall x$$

Computation duration

$$m_{x,y} \geq 0, \sum_y m_{y,x} = \frac{I_x{}^{input}}{I^{input}}, \sum_x \sum_y m_{x,y} = 1, \forall x, y$$

Placement constraint by data location

# Reduce-task Placement LP

| | |
|---|---|
| $\boldsymbol{r_x}$ | Fraction of reduce-tasks placed at site x |
| $T_{shufl}$ | Network duration for intermediate data shuffling |
| $T_{red}$ | Computation duration for reduce-stage |
| $S_x, {B_x}^{up}, {B_x}^{down}, {I_x}^{shufl}$ | #slots, up/down b/w, data volume at site x |
| $n_{red}; t_{red}$ | #reduce-tasks; duration of a reduce-task |

$$\min_{r_x} T_{shufl} + T_{red}$$

**Minimize total duration (net. + comp.)**

$$s.t.$$

$$T_{shufl} \geq \frac{{I_x}^{shufl} \times (1 - r_x)}{{B_x}^{up}}, \forall x$$

Upload transfer duration

$$T_{shufl} \geq \frac{(\sum_{y \neq x} {I_y}^{shufl}) \times r_x}{{B_x}^{down}}, \forall x$$

Download transfer duration

$$T_{red} \geq t_{red} \times \left(\frac{n_{map} \times r_x}{S_x}\right), \forall x$$

Computation duration

$$r_x \geq 0, \sum_x r_x = 1, \forall x$$

Placement constraint by data location

Challenge 1:

Heterogeneity in Resource Distribution, and Mismatch with Resource Demands

→ Reduce bottleneck of delay, and balance workloads across the sites

# Challenge 2:
# Interdependency between Task Placement and Job Scheduling

# Job Scheduling Example

- ***3*** slots per site; ***1GBps*** upload/download bandwidth
- ***100MB*** data per task; ***1s*** computation time per task

| | Job A Placement; Response Time | Job B Placement; Response Time | Average Response Time |
|---|---|---|---|
| Ideal Placement: run exclusively | **(0,1,2) → 1s** | **(2,4,6) → 2s** | **1.5s** |
| Run job A first, then job B | **(0,1,2) → 1s** | **(6,4,2) → 2.4s** | **1.7s** |
| Run job B first, then job A | **(3,0,0) → 2.3s** | **(2,4,6) → 2s** | **2.15s** |

- Not all jobs get ideal placement in optimal schedule
- Complex interaction between job scheduling and task placement

# Job Scheduling Solution

- ***Decouple job scheduling and task placement***
- Job scheduling
  - Schedule ***faster*** jobs first to **reduce waiting time** (SJF)
  - Jobs' durations estimated by task placement model
- Task placement
  - Solve task placement model based on remaining network/compute capacity to **minimize computation time**
  - Remaining capacity determined by job order

- Faster jobs get as much resource as possible
  - Other jobs may starve…

# Incorporating Fairness Scheduling

- A control knob **ε** $(0 \leq \varepsilon \leq 1)$ balancing ***fairness*** and ***response time***
  - Each job receives *at least* $(1-\varepsilon)*(\frac{f_i}{\sum_i f_i})$ slots
    - $\boldsymbol{f_i}$ is job i's remaining number of tasks
  - The number of slots one job can get is capped
    - Total #slots - #reserved slots
  - ε→ 0, completely fairness oriented
  - ε→ 1, completely response time oriented

Challenge 2:

Interdependency between

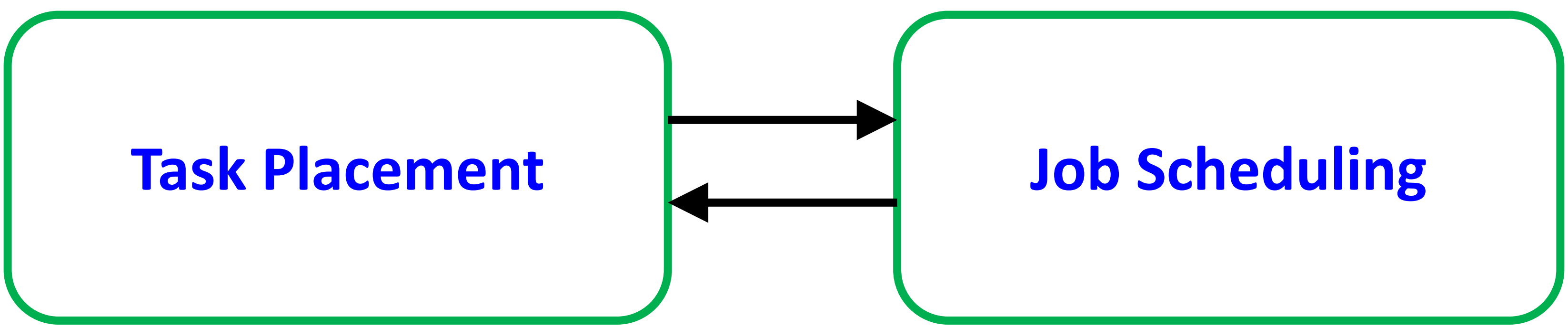
Task Placement and Job Scheduling

→ Decouple and solve iteratively

# *Tetrium:* Design Summary

**Scheduling Instance**

Instantiation: arrival of the available slots, arrival of the new job
Input: current jobs, currently available slot distribution, network bandwidth
Termination: once all slots are allocated, or all jobs are allocated with slots




***Minimize the jobs' response times!***

# Prototype and Evaluation

- Tetrium **prototype** on top of Spark
  - Inject job scheduling and task placement into scheduler
  - Estimate task running time based on peer tasks
  - Batch available slots to reduce scheduling fluctuation
  - Solve LP optimizations with Gurobi Solver

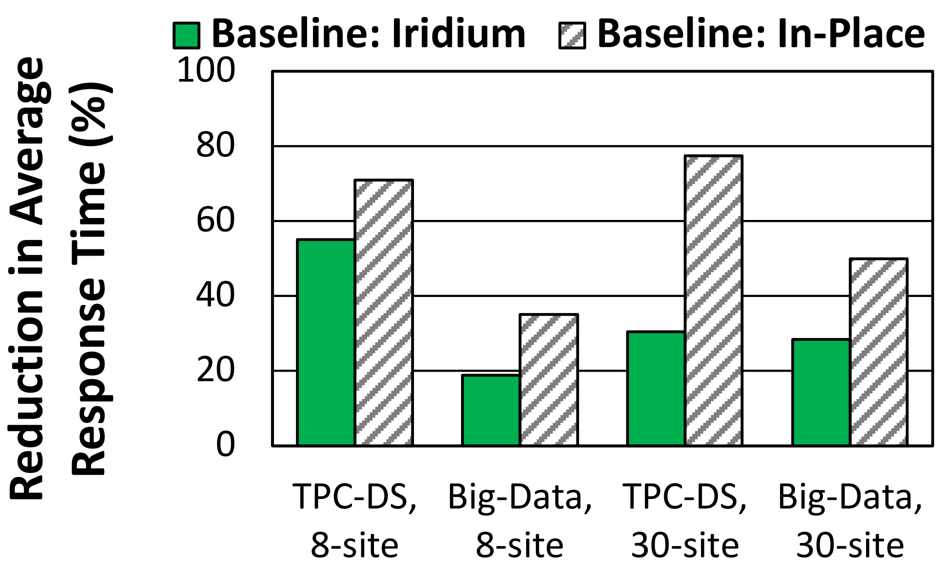
- Tetrium **deployment** in geo-distributed EC2 cluster
  - TPC-DS (6~16 stages) and Big Data (2~5 stages) Benchmark

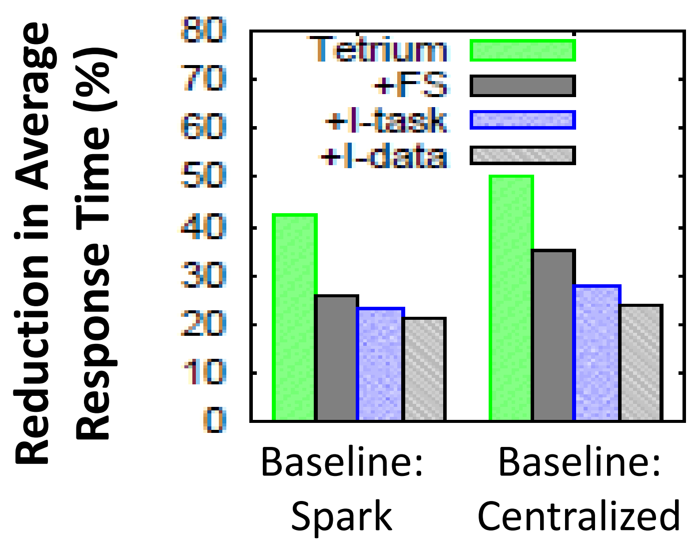
- Performance characterization through large-scale trace-driven **simulations**
  - Traces of 3000-machine production cluster

# Performance Improvements

Reduction in average job response time compared to baselines

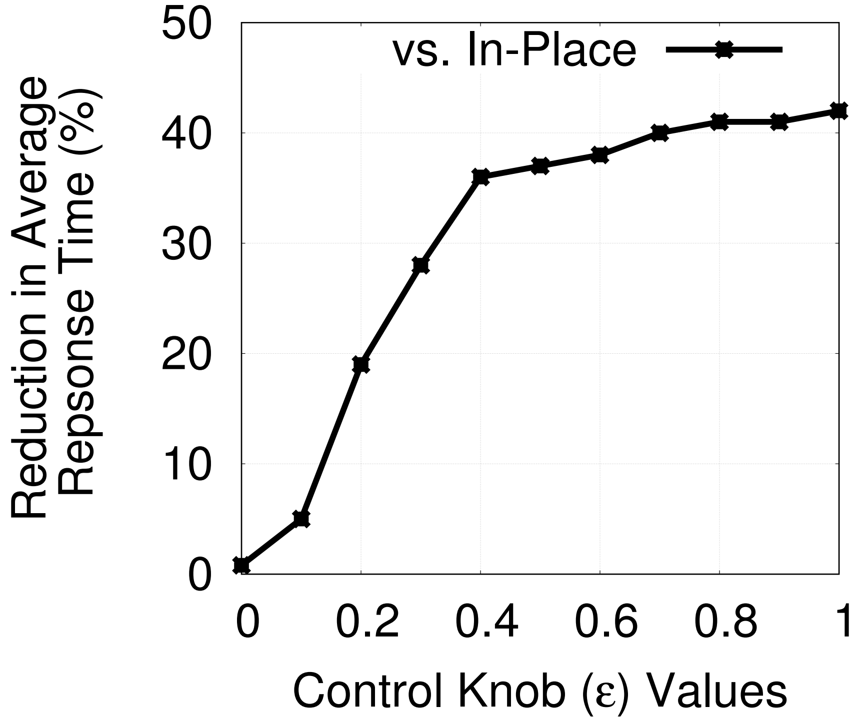
**In-Place (Spark):** in-place for task placement; fair scheduling across jobs
**Iridium:** network-centric for task placement; fair scheduling across jobs
**Centralized:** aggregate all data to one power site




- Gains are up to **77%** and **55%** compared to **In-Place** and **Iridium**
- Gains are higher with more sites or with more workloads
- Gains attribute to both job scheduling and task placement

# Response Time vs. Fairness




- ***Comparable gains in response time*** even when each jobs is guaranteed to be allocated ***60%*** of the proportional slots

# Other Key Results

- Gains are universal across all job sizes
  - 50% (36%) improvements for large (small) jobs
- Intermediate-input data ratio
  - More improvements for higher ratio
- Scheduling overhead
  - Scheduling decision ~1s;  LP optimization solving ~100ms
  - Keep overhead low by focusing on the faster jobs

*Thank you!*