A Case for Performance-Centric Network Allocation

Gautam Kumar, Mosharaf Chowdhury,
Sylvia Ratnasamy, Ion Stoica
Datacenter Applications
Data Parallelism

• Applications execute in several computation stages and require transfer of data between these stages (communication).

• Computation in a stage is split across multiple nodes.

• Network has an important role to play, 33% of the running time in Facebook traces. (Orchestra, SIGCOMM 2011)
Data Parallelism

• Users, often, do not know what network support they require.
  ▪ Final execution graph created by the framework.

• Frameworks know more, provide certain communication primitives.
  ▪ e.g., Shuffle, Broadcast etc.
Scope

Private clusters running data parallel applications.

Little concern for adversarial behavior.

Application level inefficiencies dealt extrinsically.
Current Proposals
Explicit Accounting

• Virtual cluster based network reservations. (Oktopus, SIGCOMM 2011)

• Time-varying network reservations. (SIGCOMM 2012)

DRAWBACK:

Exact network requirements often not known; non work-conserving.
Fairness-Centric

- Flow level fairness or Per-Flow. (TCP)
- Fairness with respect to the sources. (Seawall, NSDI 2012)
- Proportionality in terms of total number of VMs. (FairCloud, SIGCOMM 2012)

DRAWBACK:
Gives little guidance to developers about the performance they can expect while scaling their applications.
In this work . . .

• A new perspective to share the network amongst data-parallel applications – performance-centric allocations:
  ▪ enabling users to reason about the performance of their applications when they scale them up.
  ▪ enabling applications to effectively parallelize to preserve the intuitive mapping between scale-up and speed-up.

• Contrast / relate performance-centric proposals with fairness-centric proposals.
Performance-Centric Allocations
Types of Transfers*

Shuffle

Broadcast

(*Orchestra, SIGCOMM 2011)
Scaling up the application

Shuffle

Total Data = 2λ

Broadcast

Total Data = 4λ

2X Scale UP

λ

λ/2

λ/2

HotCloud

June 12, 2012
Performance-Centric Allocations

• Understand the support that the application needs from the network to effectively parallelize.

• At a sweet spot – framework knows application’s network requirements.
Shuffle-only clusters
Shuffle-only Clusters

Per-Flow

$A_m \xrightarrow{2\lambda} \alpha \xrightarrow{} A_r$

$t_{A_m}$

$B_m \xrightarrow{\lambda/2} \alpha \xrightarrow{} B_r$

$t_{A_m}/2$

$t_{B_s} = \lambda/2\alpha = t_{A_s}/4$

$t_{A_r}/2$

$t_{B} < t_{A}/2$

Proportional

$A_m \xrightarrow{2\lambda} \alpha \xrightarrow{} A_r$

$t_{A_m}$

$B_m \xrightarrow{\lambda/2} \alpha/2 \xrightarrow{} B_r$

$t_{A_m}/2$

$t_{B_s} = \lambda/\alpha = t_{A_s}/2$

$t_{A_r}/2$

$t_{B} = t_{A}/2$
Broadcast-only Clusters

\[
\begin{align*}
\text{Am} & \xrightarrow{2\lambda} \text{Ar} \\
\text{Bm} & \xrightarrow{\lambda} \text{Br} \\
\text{Bm} & \xrightarrow{\lambda} \text{Bm}
\end{align*}
\]
Broadcast-only Clusters

Proportional

- $A_m \xrightarrow{2\lambda} t_{A_m}$
- $B_m \xrightarrow{\lambda} t_{B_m}/2$
- $A_r \leftarrow \alpha t_{A_s} = 2\lambda/\alpha$
- $B_r \leftarrow \alpha/2$
- $t_B > t_A/2$

- $t_{B_s} = 2\lambda/\alpha = t_{A_s}$
- $t_{A_r} = t_{A_r}$

Per-Flow

- $A_m \xrightarrow{2\lambda} t_{A_m}$
- $B_m \xrightarrow{\lambda} t_{B_m}/2$
- $A_r \leftarrow \alpha t_{A_s} = 2\lambda/\alpha$
- $B_r \leftarrow \alpha/2$
- $t_B = t_A/2$

- $t_{B_s} = \lambda/\alpha = t_{A_s}/2$
- $t_{A_r} = t_{A_r}$

HotCloud
June 12, 2012
Recap

- TCP in shuffle gives more than requisite speed-up and thus hurts performance of small jobs. Proportionality achieves the right balance.

- Proportionality in broadcast limits parallelism. TCP achieves the right balance.
Complexity of a transfer

• $x_N$-transfer if $x$ is the factor by which the amount of data transferred increases when a scale up of $N$ is done, $x \in [1, N]$.

• Shuffle is a $1_N$-transfer and broadcast is an $N_N$-transfer.

• Performance-centric allocations encompass $x$. 
Heterogeneous Frameworks and Congested Resources
• Share given based on the complexity of the transfer.

• The job completion time of both jobs degrades uniformly in the event of contention.
Network Parallelism

- Isolation between the speed-up due to the scale-up for the application and the performance degradation due to finite resources.

\[ y' \leftarrow (\alpha) \times \frac{y}{N} \]

- \( y' \): new running time after a scale-up of \( N \)
- \( y \): old running time
- \( \alpha \): degradation due to limited resources
Summary
• Understand **performance-centric allocations** and their relationship with **fairness-centric proposals**.
  
  ▪ **Proportionality** is the performance-centric approach for **shuffle-only clusters**.
  
  ▪ Breaks down for broadcasts, **per-flow** is the performance-centric approach for **broadcast-only clusters**.

• An attempt to a performance-centric proposal for **heterogeneous transfers**.
  
  ▪ Understand what happens when resources get **congested**.
Future Work
• A more rigorous formulation.
  ▪ Some questions to be answered: different $N_1$ and $N_2$ on both sides of the stage etc.

• Analytical and experimental evaluation of the policies.
  ▪ Whether redistribution of completion time or total savings.
Thank you