

Assigning Tasks for Efficiency in Hadoop

Michael J. Fischer¹, Xueyuan Su¹, and Yitong Yin²

¹Department of Computer Science
Yale University, U.S.A

²State Key Laboratory for Novel Software Technology
Nanjing University, China

For Presentation at ACM SPAA'10
June 13, 2010



Outline

- 1 Data-Intensive Distributed Computing
 - Background
 - Framework
- 2 The Task Assignment Problem
 - A Simplified Model of Hadoop
 - The Cost Function and Server Load
- 3 HTA Decision Problem
- 4 Feasible Approximation Algorithms



Outline

- 1 Data-Intensive Distributed Computing
 - Background
 - Framework
- 2 The Task Assignment Problem
 - A Simplified Model of Hadoop
 - The Cost Function and Server Load
- 3 HTA Decision Problem
- 4 Feasible Approximation Algorithms



Data-Intensive Computing

- **Challenge:**
 - New applications require storing and processing **vast amounts of data** (search/transactions/scientific computing/etc...).
 - Need a simple way to write efficient and reliable application programs.
- **Solution:** Parallelism!



MapReduce and Hadoop

- Google's MapReduce has emerged as a leading large-scale data processing architecture.
- Apache Hadoop is a free Java implementation of MapReduce in the open source software community.
- Built on large distributed platforms of commodity PC's.
- Both data storage and computation are distributed across multiple servers, often in multiple data centers.



Distributed Storage

- **Challenge:** How to store data on large collections of widely dispersed disks?
- **Solution:** Hadoop distributed file system (HDFS)
 - Files are split into huge **equal-sized** blocks.
 - Each data block is **replicated** on multiple servers.



Sequential and Parallel Computation

- **Challenge:** How to take advantage of the large number of servers available in a cluster?
- **Solution:** Break jobs down into small independent pieces that can be processed in parallel.
 - A job is split into many small **tasks**.
 - Each task processes **a single** input data block.
 - Tasks are assigned to and run on **servers**.
 - Tasks assigned to the same server run **sequentially**, while those assigned to different servers run **in parallel**.



Scheduling Tasks to Servers

- **Challenge:** How to schedule tasks among servers to minimize job completion time?
- **Goal:** Somehow assign tasks to servers to simultaneously balance load and minimize cost of remote data access.

The rest of this talk addresses the problem of achieving this goal.



Outline

- 1 Data-Intensive Distributed Computing
 - Background
 - Framework
- 2 The Task Assignment Problem
 - A Simplified Model of Hadoop
 - The Cost Function and Server Load
- 3 HTA Decision Problem
- 4 Feasible Approximation Algorithms

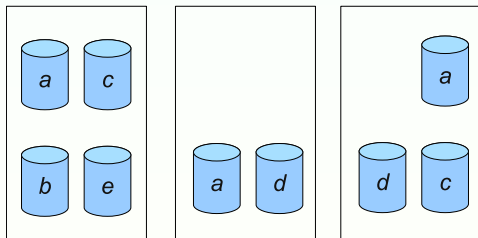
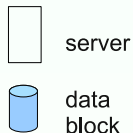


Components of the Model

- A set of **tasks** and a set of **data blocks**. Each data block has a corresponding task to be performed on it.
- A set of **servers** that store data blocks and perform tasks.
- A **data placement** that maps a data block to the set of servers on which **replicas** of that block are stored.
- A **task assignment** that specifies the server on which a task will run.
- A **cost function** that determines the completion time of the system.



Data Placement on Servers



Server 1

Server 2

Server 3

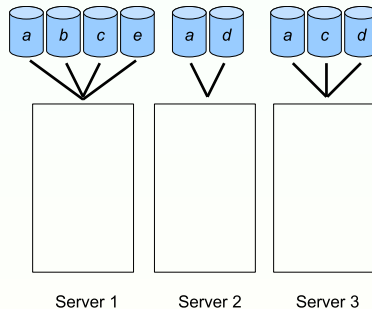
Every data block occurs on at least one server.

Block	#Replicas
<i>a</i>	3
<i>b</i>	1
<i>c</i>	2
<i>d</i>	2
<i>e</i>	1



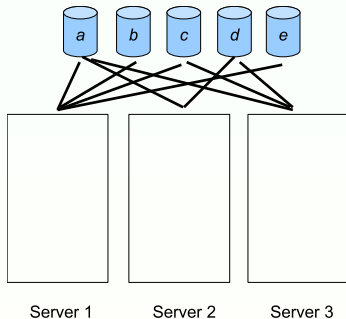
Data Placement Graph

Slide data blocks up out of the server rectangles and connect them by lines. This leads to a bipartite graph where each replica has an edge to its corresponding server.

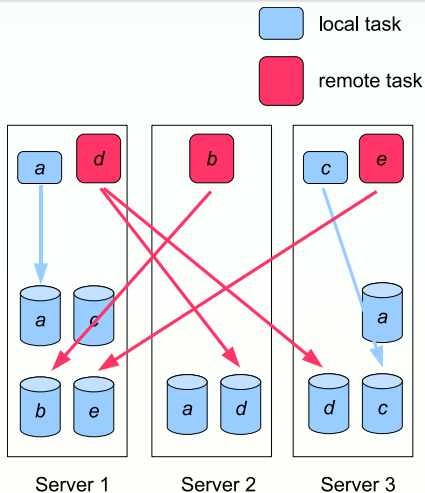


Data Placement Graph (cont.)

Merge the replica nodes with the same block label together, giving the bipartite graph called the **data placement graph** G_ρ .



Assigning Tasks to Servers



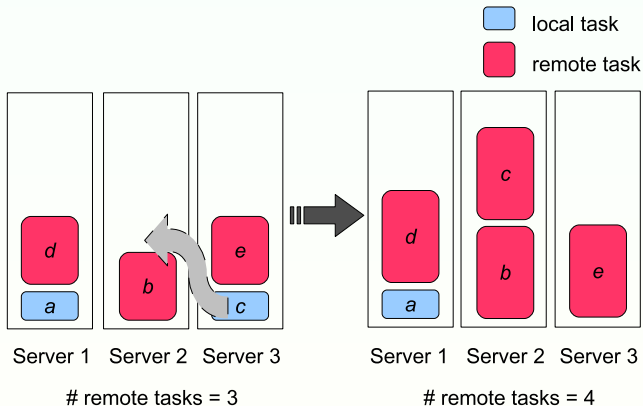
remote tasks = 3

- A task is **local** if it is on the same server as its corresponding data block. Its cost is w_{loc} .
- A task is **remote** if it is on a different server from its corresponding data block. Its cost is w_{rem} .
- We assume $w_{rem} \geq w_{loc}$.



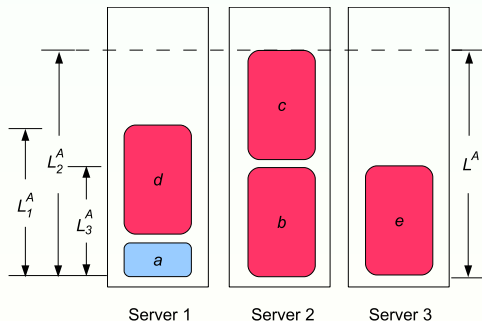
Remote Cost

w_{rem} increases with the total number of remote tasks.
This models increased latency due to network congestion.



Server Load and Maximum Load

- The load L_s^A of a server s under assignment A is the sum of costs introduced by all tasks that A assigns to s .
- The maximum load L^A of assignment A is the maximum server load under A . In the following example, $L^A = L_2^A$.



HTA Problems

Definition

The *HTA Optimization Problem* is, given a Hadoop system, to find a task assignment that **minimizes** the **maximum load**.

Definition

The *HTA Decision Problem* is, given a Hadoop system and a server capacity k , to decide whether there exists a task assignment with **maximum load** $\leq k$.



Outline

- 1 Data-Intensive Distributed Computing
 - Background
 - Framework
- 2 The Task Assignment Problem
 - A Simplified Model of Hadoop
 - The Cost Function and Server Load
- 3 HTA Decision Problem
- 4 Feasible Approximation Algorithms



NP-Completeness

Theorem

The HTA decision problem is NP-complete.



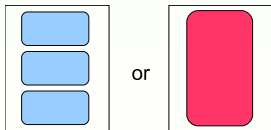
Proof Method

Given a 3CNF formula, we construct an instance of the HTA decision problem such that

- the local cost is 1,
- the remote cost is 3,
- the server capacity is 3.

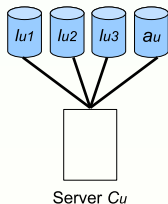
The resulting HTA instance has a task assignment of maximum load at most 3 if and only if the 3CNF formula is satisfiable.

Note that a server can be assigned at most 3 local tasks or 1 remote task.



A Piece of the Structure

- Let $C_u = (l_{u1} \vee l_{u2} \vee l_{u3})$ be a clause in the 3CNF formula and each l_{uv} be a literal.
- Construct a corresponding **clause gadget**, containing 1 clause server C_u , 3 literal tasks l_{u1}, l_{u2}, l_{u3} and 1 auxiliary task a_u .
- The server can only accept 3 local tasks, and thus 1 of the tasks needs to be assigned elsewhere.



Intuition Behind the Structure

There exists a feasible task assignment such that:

- The **auxiliary task** is assigned locally to its corresponding clause server.
- All **false literal tasks** are assigned locally to their corresponding clause servers.

Therefore, some literal task must be assigned elsewhere.

This will imply that the corresponding literal of the 3CNF formula is true.

Please refer to the original paper for more details...



Outline

- 1 Data-Intensive Distributed Computing
 - Background
 - Framework
- 2 The Task Assignment Problem
 - A Simplified Model of Hadoop
 - The Cost Function and Server Load
- 3 HTA Decision Problem
- 4 Feasible Approximation Algorithms



Approximation Algorithms

We presented and analyzed two approximation algorithms.

- A simple round robin algorithm computes assignments that deviate from the optimal by a **multiplicative factor** $(w_{\text{rem}}/w_{\text{loc}})$.
- A flow-based algorithm computes assignments that are optimal to within an **additive gap** w_{rem} .



Results for the Round Robin Algorithm

- This algorithm computes an assignment A that deviates from the optimal assignment O by a multiplicative factor, i.e.,

$$L^A \leq (w_{\text{rem}}/w_{\text{loc}})L^O$$

- A matching lower bound is proved for a class of placement graphs.
- The counter-intuitive result is shown that increasing the number of data block replicas may actually increase the maximum load of the computed task assignment as long as $w_{\text{rem}} > w_{\text{loc}}$.



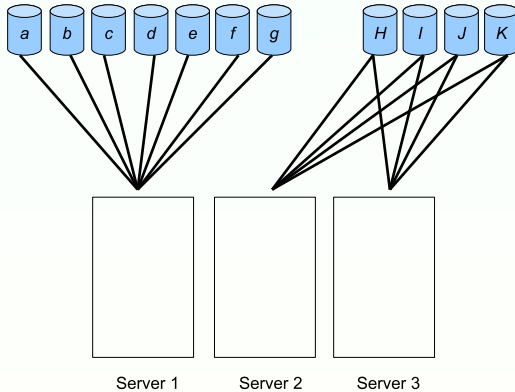
Outline of the Flow-Based Algorithm

Given a threshold τ , the algorithm runs in two phases:

- **Max-cover** uses network flow to **maximize** the number of **local tasks** such that no single server is assigned more than τ local tasks.
- **Bal-assign** finishes the assignment by repeatedly assigning unassigned tasks to a server with **minimal load**.

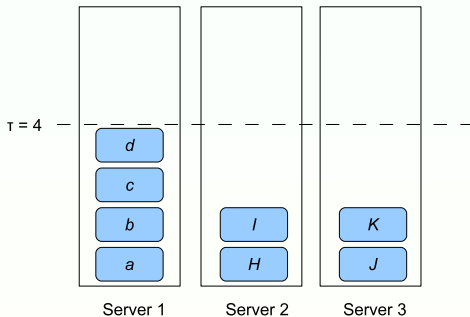


Example Data Placement



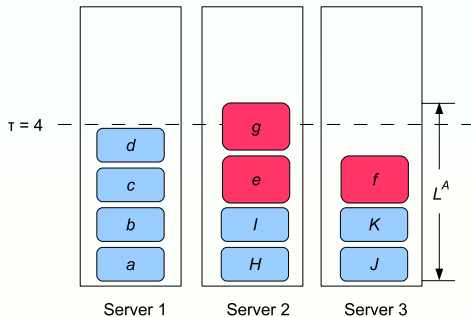
Max-cover with Threshold $\tau = 4$

Max-cover assigns each server the maximum number of local tasks, subject to the threshold $\tau = 4$.



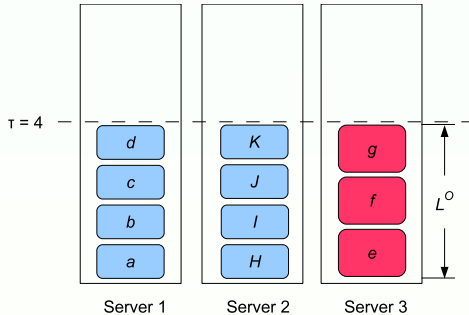
Bal-assign

Bal-assign finishes the task assignment by repeatedly assigning unassigned tasks to a server with minimal load.



Optimal Assignment O

However, optimal assignment can do better...



The Approximation Gap

Let m be the number of tasks. The algorithm tries all possible threshold, $1 \leq \tau \leq m$, and outputs an assignment with the smallest maximum load out of all the computed assignments.

Theorem

Let n be the number of servers. For $n \geq 2$, the algorithm computes an assignment A such that $L^A \leq L^O + \left(1 - \frac{1}{n-1}\right) w_{\text{rem}}$.



Time Complexity

The correctness of the algorithm does not depend on the order of choosing different values of τ . However, the running time is improved by letting max-cover at iteration $(\tau + 1)$ take as input the output of max-cover at iteration τ .

Theorem

Let m and n be the number of tasks and servers, respectively. The algorithm runs in time $O(m^2 n)$.



Time Complexity (cont.)

In practice, the placement graph is usually sparse, and the number of edges in the placement graph is $O(m + n)$.

In this case, the bal-assign phase becomes the major contributor to the algorithm time complexity. We implement bal-assign by a priority queue. Each iteration takes time $O(m \log n)$. Hence, m iterations takes total time $O(m^2 \log n)$.



Summary

- We presented a model for studying the problem of assigning tasks to servers in Hadoop systems so as to minimize job completion time.
- We showed the HTA decision problem to be NP-complete.
- We presented and analyzed two approximation algorithms.
 - A round robin algorithm that deviates from the optimal by a multiplicative factor $(w_{\text{rem}}/w_{\text{loc}})$.
 - A flow-based algorithm that achieves an additive approximation gap that is less than w_{rem} .



Future Work

The interaction between data placement and task assignment is important. It would be interesting to investigate algorithms for replica placement.



Thank you!
&
Questions?

