Distributed Offline Load Balancing in MapReduce Networks

Themistoklis Charalambous[⋆], Evangelia Kalyvianaki[‡], Christoforos N. Hadjicostis[♯], and Mikael Johansson[⋆]

[⋆] Department of Automatic Control, Royal Institute of Technology (KTH)

[‡] School of Informatics, City University London

[♯]Department of Electrical and Computer Engineering, University of Cyprus

IEEE Conference on Decision and Control (CDC '13)
Florence, Italy, December 2013

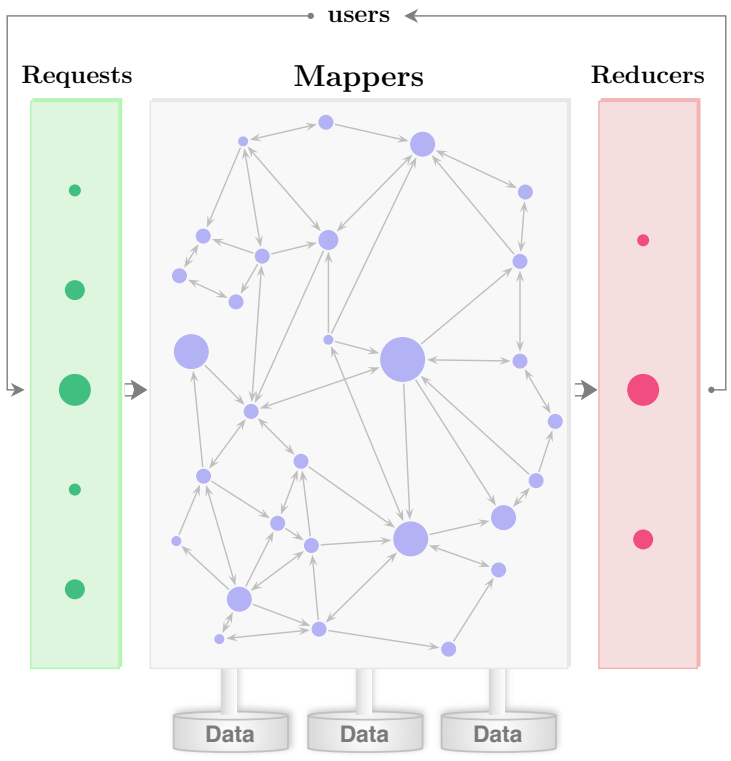
Outline

- Motivation - Introduction
- Notation and mathematical preliminaries
- Average consensus in digraphs
- Ratio consensus
- Examples
- Concluding remarks and future directions

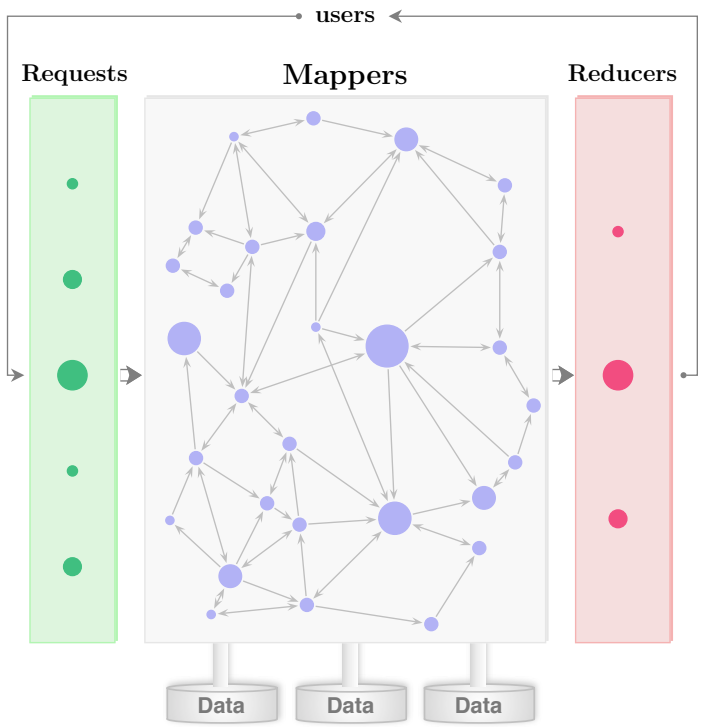
- Cloud computing enables quick application deployment over a *large pool of commodities* with *little capital investment*
- fast emerging Cloud application: big data computing
- modern applications in this domain employ: MapReduce paradigm, e.g.,
  - **Google** uses MapReduce for web search indexing, Google News and Google Maps
  - **Facebook** uses MapReduce for data warehousing operations
- MapReduce - designed to achieve high-job completion speedups through parallelism

Introduction
What is the MapReduce paradigm? (1)

- Input data divided into smaller chunks
- Input data transformed into key/value pairs
- Processing distributed to a network of processing elements
- Input data processed by two successively executed phases:
  1. Map phase: input data chunks processed independently by mappers - produce a new list of key/value pairs
  2. Reduce phase: pairs with common keys generated in the *map* phase are merged into a final list of key/values pairs



Introduction
What is the MapReduce paradigm? (2)



- Different mappers and different reducers can be executed in parallel.
- Scalability achieved by distributing the workload
  1) within the network of mappers
  2) within the network of reducers;

Motivation
Stragglers and previous approaches

- Stragglers: take $\geq$ 1.5 times the median task in that phase, due to
  - contention for resources
  - heterogeneity (e.g., disk loss rate)
  - mapReduce artifacts
- Current schemes:
  - replicate the execution of data processed by stragglers using back-up [Dean & Ghemawat, 2004], [Zaharia *et al*, 2008]
  - combine specialized solutions based on root-cause analysis of outliers, network-aware placements, and duplication [Ananthanarayanan *et al*, 2010]
  - rely on centralized solutions to map tasks or data to nodes prior to application execution [Xie *et al*, 2010], [Verma *et al*, 2011]

Q: How can we deal with stragglers in a distributed fashion?

Contribution

- **Target:**
  - divide the set of input data into *variable-sized* chunks
  - assign each chunk with a mapper process $\rightarrow$ all mappers commence and complete processing with no/small differences
  - we address the problem of stragglers due to data processing load imbalances in heterogeneous networks

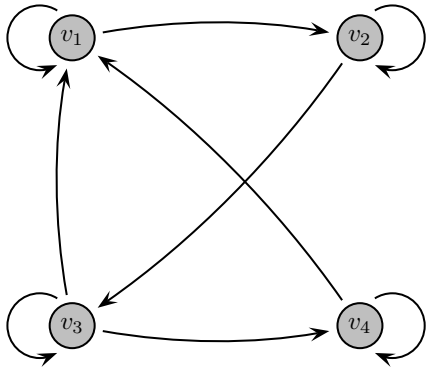
- **Propose:** a distributed algorithm, robustified ratio consensus
  - such that each mapper finishes its allocated workload at approximately the same time as the other mappers
  - allows mappers on heterogeneous nodes to operate asynchronously
  - operates in directed networks
  - mappers converge to proportional workload balancing between them
  - required limited local information exchanges

Contribution

Advantages over other methods (e.g., [Gonzalez-Ruiz *et al*, 2009])

- Lifted the assumption: cluster nodes need to be homogeneous and have the same resource capacity
  - allow for heterogeneous nodes
  - load distributed according to nodes' capacities (in terms of CPU, memory, disk space, etc.)
  - network topology might be weighted due to different bandwidth capabilities or time-varying delays
- Lifted the assumption: link-level delay needs to be either negligible or synchronous
  - our algorithm exhibits asymptotic converge in the presence of bounded delays
- Lifted the assumption: network is undirected
  - we consider directed graphs (digraphs)
  - In practice, network topology might become loosely directed because of delays, packet losses and asymmetric links
  - directed graph structure reduces overhead communication and can admit faster convergence

- Distributed systems conveniently captured by digraphs
  1. Components represented by vertices (nodes)
  2. Communication and sensing links represented by edges



- Consider a network with nodes $(v_1, v_2, \ldots, v_N)$ (in this case it is the network of mappers)
- Nodes can receive information according to (possibly directed) communication links
- Each node v_j has some initial value $\pi_j[0]$ of the amount of resources that it intends to contribute towards the map phase and a maximum capacity $\pi_j^{\max}$ that can provide

Graph Notation

- *Digraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - Nodes (system components) $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$
  - Edges (directed communication links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ where $(v_j, v_i) \in \mathcal{E}$ iff node v_j can receive information from node v_i
  - In-neighbors $\mathcal{N}_j^- = \{v_i \mid (v_j, v_i) \in \mathcal{E}\}$; in-degree $\mathcal{D}_j^- = |\mathcal{N}_j^-|$
  - Out-neighbors $\mathcal{N}_j^+ = \{v_l \mid (v_l, v_j) \in \mathcal{E}\}$; out-degree $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$
- *Adjacency matrix A:* $A(j, i) = 1$ if $(v_j, v_i) \in \mathcal{E}$; $A(j, i) = 0$ otherwise
- *Undirected graph:* $(v_j, v_i) \in \mathcal{E}$ iff $(v_i, v_j) \in \mathcal{E}$ (bidirectional links)
  In undirected graphs, we have (for each node j)
  $\mathcal{N}_j^+ = \mathcal{N}_j^-$ and $\mathcal{D}_j^+ = \mathcal{D}_j^- = \mathcal{D}_j$; also, $A = A^T$
- *(Strongly) connected (di)graph* if for any $i, j \in \mathcal{V}$, $j \neq i$, there exists a (directed) path connecting them, i.e.,

$$v_i = v_{i_0} \to v_{i_1}, \; v_{i_1} \to v_{i_2}, \; \ldots, \; v_{i_{t-1}} \to v_{i_t} = v_j$$

- Average Consensus: All nodes calculate (in a distributed manner) the average $\frac{1}{N}\sum_{\ell=1}^{N} x_\ell[0]$ of initial values
- Consider linear iterations of the form

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{i\in\mathcal{N}_j^-} p_{ji}x_i[k] \ , \ \forall j \in \{1,2,...,N\}$$

  where p_{ji} are constant weights and $x_j[0]$ is the initial value of node j
- Can be written in compact form as

$$\begin{aligned} x[k+1] &= Px[k] \\ x[0] &= [x_1[0]\ x_2[0]\ \dots\ x_N[0]]^T \end{aligned}$$

- Weight matrix P such that $P(j,i) = p_{ji}$
  Note: $P(j,i) = 0$ if $(j,i) \notin \mathcal{E}$

Conditions for Asymptotic Average Consensus

- Necessary and sufficient conditions on P for asymptotic average consensus [Xiao & Boyd, 2004]
  1. P has a simple eigenvalue at 1 with left eigenvector $\mathbf{1}^T = [1\ 1\ \ldots\ 1]$ and right eigenvector $\mathbf{1} = [1\ 1\ \ldots\ 1]^T$
  2. All other eigenvalues of P have magnitude strictly smaller than 1
- As $k \to \infty$, $P^k \to \frac{1}{N}\mathbf{1}\mathbf{1}^T$ which implies that

$$\lim_{k\to\infty} x[k] = \frac{1}{N}\mathbf{1}\mathbf{1}^T x[0] = \left(\frac{\sum_{\ell=1}^{N} x_\ell[0]}{N}\right)\mathbf{1} \equiv \overline{x}\mathbf{1}$$

- Nonnegative $p_{ji} \Longrightarrow P$ is primitive bistochastic

How to distributively reach the average in digraphs?

Average Consensus in Digraphs

- In undirected graphs nodes have a number of ways to distributively choose their weights so as to form a bistochastic matrix
- Digraphs not as easy to handle, even in a centralized manner (because in general $\mathcal{D}_j^+ \neq \mathcal{D}_j^-$)
- Approaches:
  - Distributed algorithms to obtain weights that form bistochastic matrices [Gharesifard & Cortés, 2012], [T.C. & C.N.H., 2013]
  - Distributed approaches that introduce additional state variables and use broadcast gossip [Franceschelli *et al*, 2011], [Cai & Ishii, 2011] and reach the average consensus *asymptotically*
  - Run two coupled iterations simultaneously (ratio consensus) [Benezit *et al*, 2010], [A.D. Domíngez-García & C.N.H., 2010], [C.N.H. & T.C., 2011] that reach the average consensus *asymptotically*

Average Consensus using Ratio Consensus

Pair of Simultaneous Linear Iterations

- Run two iterations [Benezit *et al*, 2010], [D-G & H, 2010]

$$\begin{array}{rcl} \pi[k+1] & = & P_c\pi[k] \\ \pi[0] & = & [\pi_1[0]\dots\pi_N[0]]^T \end{array} \quad \Bigg| \quad \begin{array}{rcl} x[k+1] & = & P_c x[k] \\ x[0] & = & \mathbf{1} \end{array}$$

- Matrix P_c st $P_c(l,j) = \frac{1}{1+\mathcal{D}_j^+}$ for $v_l \in \mathcal{N}_j^+$ (zero otherwise)
- Since P_c is primitive column stochastic, we know that as $k \to \infty$, $P_c^k \to \mathbf{v}\mathbf{1}^T$ for a *strictly positive* vector $\mathbf{v}$ such that $\mathbf{v} = P_c\mathbf{v}$ ($\mathbf{v}$ is *normalized* so that its entries sum to unity)
- This implies that

$$\begin{array}{rcl} \lim_{k\to\infty} \pi[k] & = & \mathbf{v}\mathbf{1}^T\pi[0] = \left(\sum_{\ell=1}^N \pi_\ell[0]\right)\mathbf{v} \\ \lim_{k\to\infty} x[k] & = & \mathbf{v}\mathbf{1}^T x[0] = N\mathbf{v} \end{array}$$

- For all nodes $j \in \{1,2,\dots,N\}$, ratio converges

$$\frac{\pi_j[k]}{x_j[k]} \to \frac{v_j\sum_{\ell=1}^N \pi_\ell[0]}{v_j N} = \frac{\sum_{\ell=1}^N \pi_\ell[0]}{N} \equiv \overline{\pi}$$

Robustified Ratio Consensus
in the presence of bounded time-varying delays

- A transmission on link (v_j, v_i) (from node v_i to node v_j) at any time-step k can be delayed by 0, 1, ..., D steps (arriving at node v_j respectively at time-step $k+1$, $k+2$, ..., $k+D+1$)
- Unknown time-dependent delay function for each link (v_j, v_i)

$$delay_{(j,i)}[k] = d \ , \quad d \in \{0, 1, 2, ..., D\}$$

- $delay_{(j,j)}[k] = 0$ for all k and $v_j \in \mathcal{V}$
(own value is always available)
- Ratio consensus can be adopted to work in the presence of bounded time-varying delays [C.N.H. & T.C., 2011]
- Can also handle time-varying topologies (as long as nodes know their out-degrees) [C.N.H. & T.C., 2012]

- A new MapReduce job d needs to process a set of input data
- Total demand of resources required by d in the map phase is given by ρ_d.
- ρ_d can be approximately estimated via offline application profiling
- Problem to be solved:

$$\rho_d = \sum_{v_\ell \in \mathcal{V}} \pi_\ell[m],$$

such that

$$0 \leq \pi_j[m] \leq \pi_j^{\max}, \ \ \forall v_j \in \mathcal{V}$$

$$0 \leq \rho_d \leq \sum_{v_\ell \in \mathcal{V}} \pi_\ell^{\max} \triangleq \chi^{\max}$$

- A simple, feasible solution $\pi^\dagger$

$$\pi_j^\dagger = \frac{\rho_d}{\chi^{\max}} \pi_j^{\max}, \ \forall v_j \in \mathcal{V}$$

- Each node v_j updates its demanded amount $\pi_j[k]$ by combining it with the available (possibly delayed) demanded amount of its neighbors ($\pi_i[s]$ $s \in \mathbb{Z}, s \leq k,\ v_i \in \mathcal{N}_j^-$)

$$\pi_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\pi_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} \frac{1}{1+\mathcal{D}_i^+}\pi_i[k-r]I_{k-r,ji}[r]$$

where

$$I_{k,ji}(\tau) = \begin{cases} 1, & \text{if } \tau_{ji}[k] = \tau, \\ 0, & \text{otherwise.} \end{cases}$$

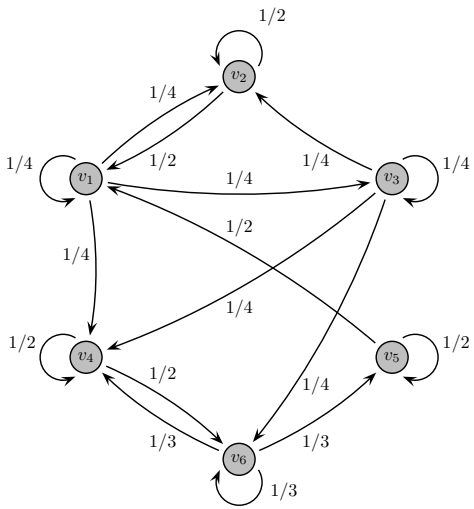
- Run two iterations, (with $y_j[0] = \pi_j[0]$ and $y_j[0] = \pi_j^{\max}$),

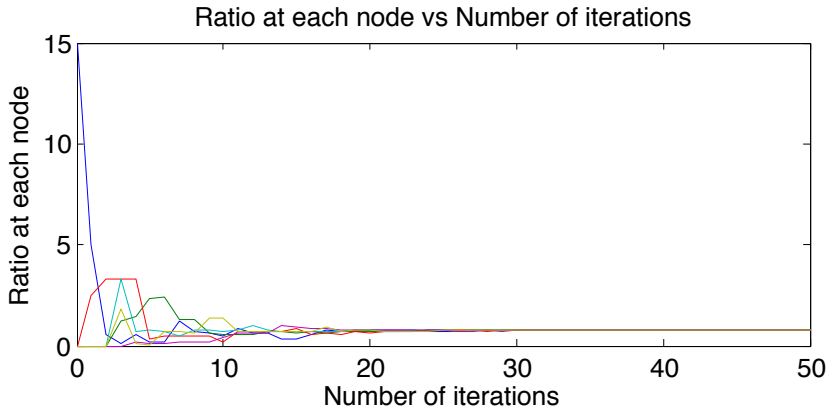
$$\lim_{k\to\infty} \mu_j[k] = \frac{\sum_{v_\ell \in \mathcal{V}} \pi_\ell[0]}{\sum_{v_\ell \in \mathcal{V}} \pi_\ell^{\max}[0]} = \frac{\rho_d}{\chi^{\max}}, \quad \forall v_j \in \mathcal{V}$$

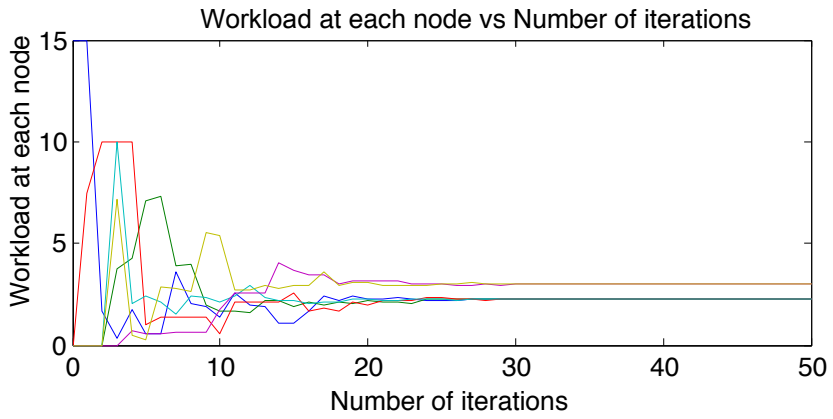
- Node v_j can obtain the amount of its load as

$$\lim_{k\to\infty} \mu_j[k]\pi_j^{\max} = \frac{\rho_d}{\chi^{\max}}\pi_j^{\max}$$

- The capacity of the mappers in this network is given by $\pi^{\max}[0] = (3 \;\; 3 \;\; 3 \;\; 3 \;\; 4 \;\; 4)^T$



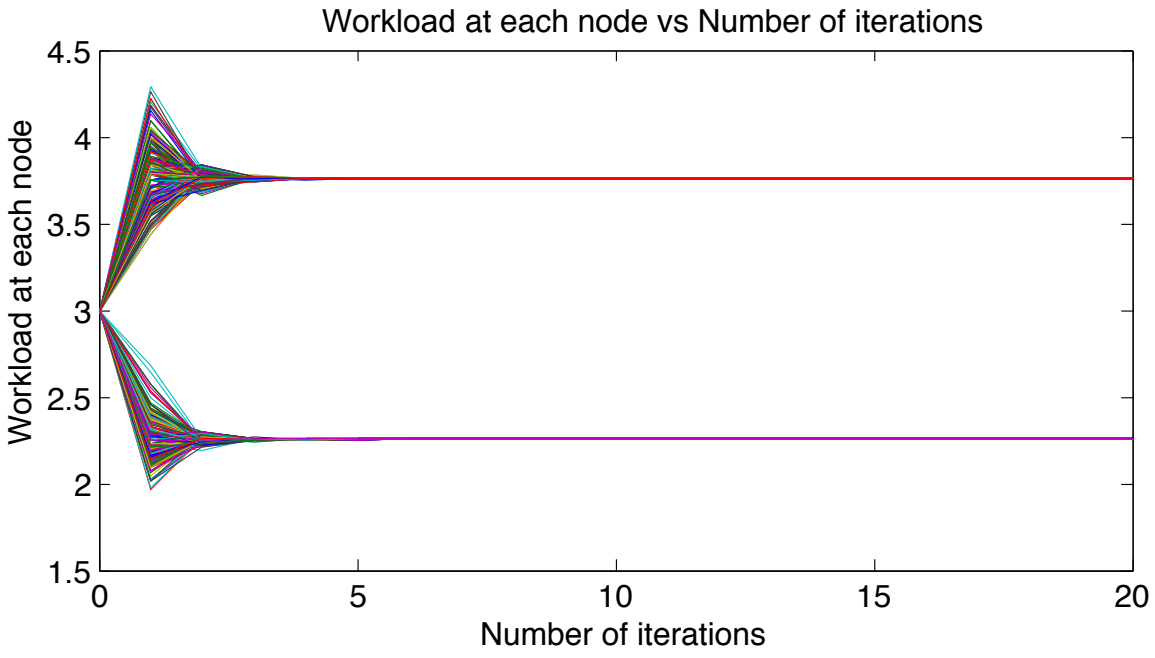




- the demanded workload amount injected into the network of mappers is given by $\pi[0] = (15 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0)^T$

- The amount of workload offered in the network by each node in the presence of asymmetric time-varying delays

- We consider a set of 500 mappers each with initial workload $\pi_j[0] = 3$
- Half of them have capacity $\pi_j^{max} = 3$ and the other half $\pi_i^{max} = 5$, $i \neq j$



Concluding Remarks and Future Directions

Conclusions:

- Proposed a distributed algorithm for workload distribution in MapReduce networks of clusters
  - allows mappers on heterogeneous nodes to operate asynchronously
  - operates in directed networks
  - nodes converge to proportional workload balancing
  - has limited local information exchanges

Future work:

- Preliminary results: assumed the load can be a continuous number, whereas in reality the workload has to be quantized
- Resources needed to serve each of the map jobs may not be known *a priori* $\rightarrow$ not possible to allocate a precise fraction of the total job to each server
- Fault-tolerance capabilities for mappers to re-adapt their workload distribution in the face of machine failures within a large data center

References


[Dean & Ghemawat, 2004] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in Proceedings of the 6th Symposium on Operating Systems Design & Implementation (OSDI). ACM, 2004, pp. 137–150.


[Zaharia *et al*, 2008] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, “Improving MapReduce performance in heterogeneous environments,” in Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2008, pp. 29–42.


[Ananthanarayanan *et al*, 2010] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, “Reining in the outliers in MapReduce clusters using Mantri,” in Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2010, pp. 1–16.


[Xie *et al*, 2010] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares, and X. Qin, “Improving MapReduce performance through data place- ment in heterogeneous Hadoop clusters,” in IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW), 2010, pp. 1–9.

References


[Verma *et al*, 2011] A. Verma, L. Cherkasova, and R. H. Campbell, “ARIA: automatic resource inference and allocation for MapReduce environments,” in Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC), 2011, pp. 235–244.


[Xiao & Boyd, 2004] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging," Systems and Control Letters, vol. 53, no. 1, pp. 6–78, Sept. 2004.


[Benezit *et al*, 2010] F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, “Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in Proc. of International Symposium on Information Theory, pp. 1753–1757, 2010.


[A.D. Domínguez-García & C.N.H., 2011] A. D. Domínguez-García and C. N. Hadjicostis, “Coordination and control of distributed energy resources for provision of ancillary services,” in Proceedings of the First IEEE International Conference on Smart Grid Communications, Oct. 2010, pp. 537–542.


[C.N.H. & T.C., 2011] C. N. Hadjicostis and T. Charalambous, “Asynchronous coordination of distributed energy resources for the provisioning of ancillary services,” in Proc. of Allerton Conference on Communication, Control, and Computing, pp. 1500–1507, 2011.

References

- [Cai & Ishii, 2011] K. Cai and H. Ishii, “Quantized consensus and averaging on gossip digraphs," IEEE Transactions on Automatic Control, vol. 56, no. 9, pp. 2087–2100, Sept. 2011.
- [Franceschelli *et al*, 2011] M.Franceschelli, A. Giua, and C. Seatzu, “Distributed averaging in sensor networks based on broadcast gossip algorithms," IEEE Special Issue On Cognitive Sensor Networks, vol. 3, no. 3, pp. 808–817, 2011.
- [C.N.H. & T.C., 2012] Christoforos N. Hadjicostis and Themistoklis Charalambous, “Average Consensus in the Presence of Delays and Dynamically Changing Directed Graph Topologies”, arXiv:1210.4778, 2012.
- [Gharesifard & Cortés, 2012] B. Gharesifard and J. Cortés, “Distributed strategies for generating weight-balanced and doubly stochastic digraphs,” European Journal of Control, vol. 18, no. 6, pp. 539–557, 2012.
- [C.N.H. & T.C., 2013] Christoforos N. Hadjicostis and Themistoklis Charalambous, “Average Consensus in the Presence of Delays in Directed Graph Topologies", IEEE Transactions on Automatic Control, March 2014 (to appear).

Thank You!

Questions?

For more information:

themisc@kth.se
evangelia.kalyvianaki.1@city.ac.uk
chadjic@ucy.ac.cy
mikaelj@kth.se