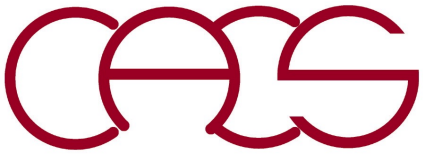
MapReduce

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Quantitative & Computational Biology
University of Southern California

Email: anakano@usc.edu

Amazon Elastic Computing Cloud (EC2)
Cloud computing support at CARC
Hadoop@USC-CARC



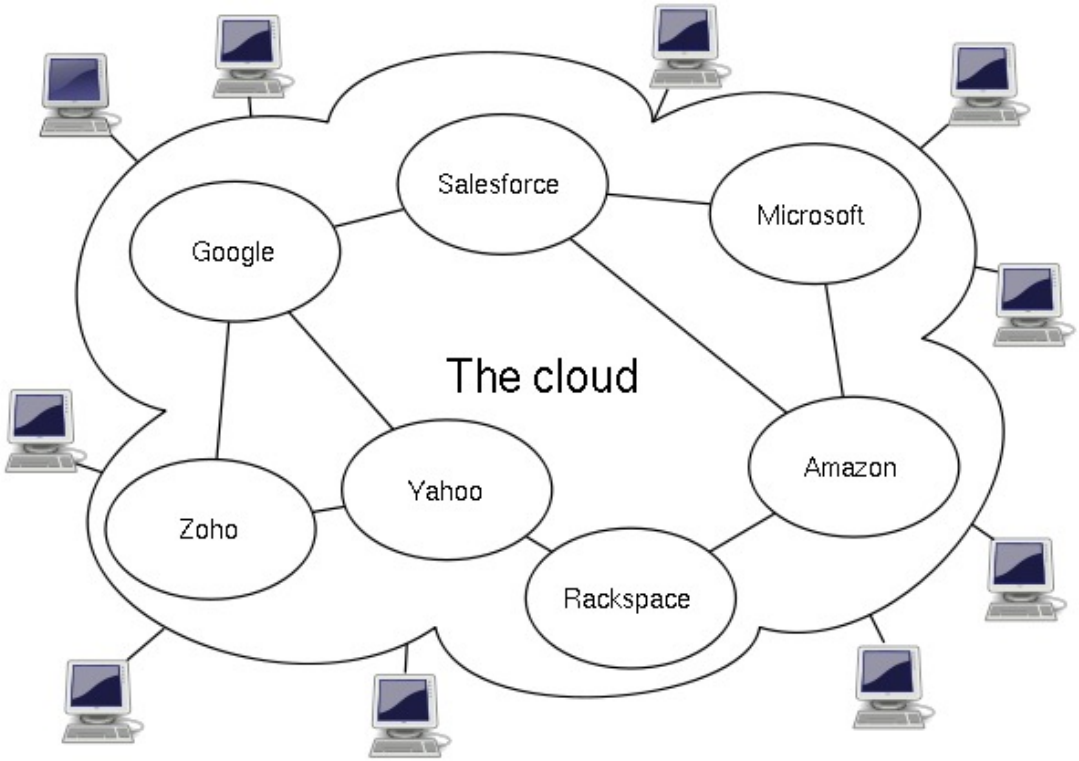
Above the Clouds: A Berkeley View of Cloud Computing

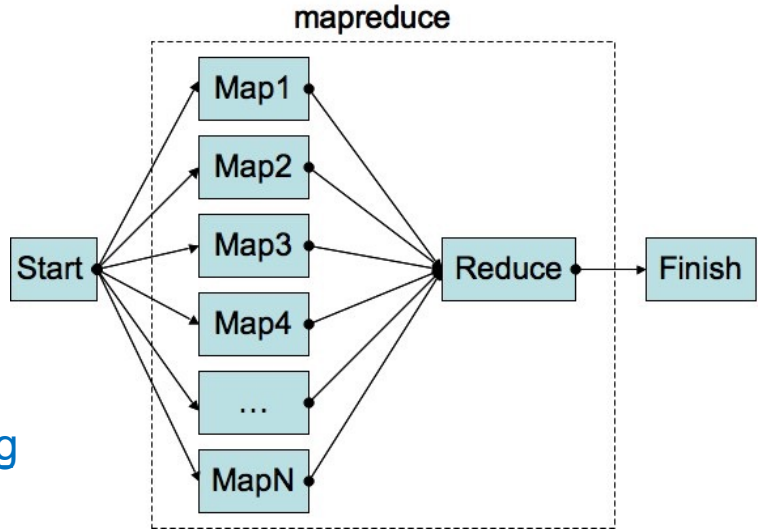
Michael Armbrust
Armando Fox
Rean Griffith
Anthony D. Joseph
Randy H. Katz
Andrew Konwinski
Gunho Lee
David A. Patterson
Ariel Rabkin
Ion Stoica
Matei Zaharia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-28
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

February 10, 2009





[Relationship between IoT, Big Data & Cloud Computing (Nick Mckanna)]

IoT: Internet of things

MapReduce

- Parallel programming model for data-intensive applications on large clusters
  - > User just implements Map() and Reduce()
- Parallel computing framework
  - > Libraries take care of everything else
    - Parallelization
    - Fault tolerance
    - Data distribution
    - Load balancing
- Developed at Google

Functional Abstraction

- **Map and Reduce functions borrowed from functional programming languages**

  (Common LISP example)

  ```
  > (mapcar '1+ '(1 2 3 4)) ⟹ (2 3 4 5)
  > (reduce '+ '(1 2 3 4)) ⟹ 10
  ```

- **Map()**
  - **Process a key/value pair to generate intermediate key/value pairs**
- **Reduce()**

  ***cf.* MPI_Allreduce()**
  - **Merge all intermediate values associated with the same key**

***cf.* Lambda function**

Lisp: `((lambda (x y) (+ x y)) 1 2)`

See https://racket-lang.org/

Example: Counting Words

- **Map()**
  - > **Input <filename, file text>**
  - > **Parses file and emits <word, count> pairs**
    - ***e.g.*** **<“hello”, 1>**
- **Reduce()**
  - > **Sums values for the same key and emits <word, TotalCount>**
    - ***e.g.*** **<“hello”, (3 5 2 7)> ⟹ <“hello”, 17>**

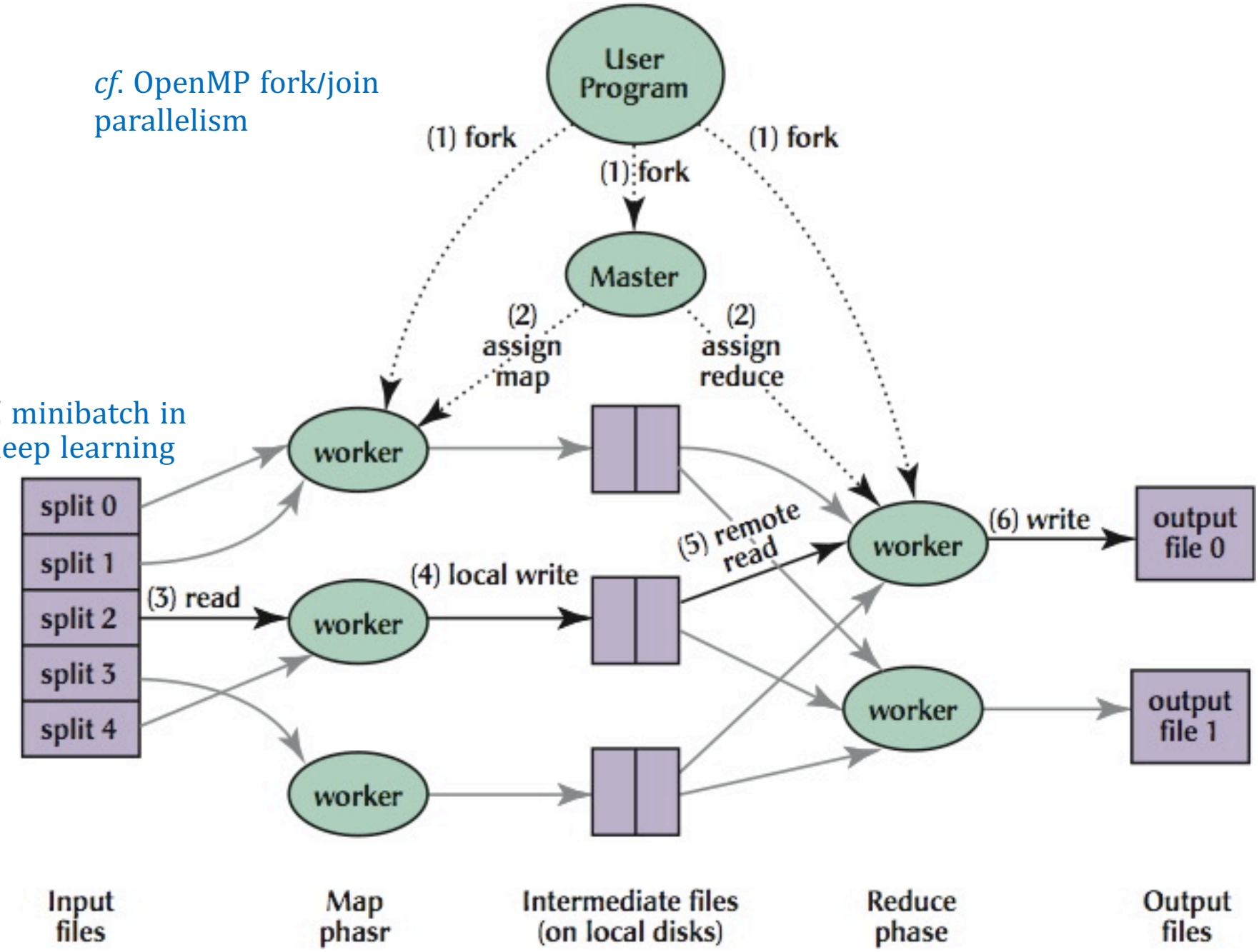
```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
  EmitIntermediate(w, “1”);

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
  result += ParseInt(v);
Emit(AsString(result));
```

Parallel Execution

cf. OpenMP fork/join parallelism

cf. minibatch in deep learning



MapReduce Resources

- **Hadoop implementation**
  `http://hadoop.apache.org`
- **MapReduce tutorial**
  `https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html`
- **Paper**
  J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, *Communications of the ACM* **51(1)**, 107 ('08)
- **Free account (Amazon Web Services)**
  `http://aws.amazon.com/free`

Cloud Supercomputing

TECHNOLOGY LAB / INFORMATION TECHNOLOGY

18 hours, $33K, and 156,314 cores: Amazon cloud HPC hits a "petaflop"

1.21 petaflops? Great scott!

by **Jon Brodkin** - Nov 12 2013, 11:00am PST

CLOUD SUPERCOMPUTING 54

USC chemistry professor Mark Thompson needed the cluster to design materials that might be well-suited to converting sunlight into solar energy.

"For any possible material, just figuring out how to synthesize it, purify it, and then analyze it typically takes a year of grad student time and hundreds of thousands of dollars in equipment, chemicals, and labor for that one molecule," Cycle Computing CEO Jason Stowe wrote in a blog post today.

Instead of doing that, Thompson uses simulation software made by Schrödinger. With that software running on Amazon, Thompson was able to simulate 205,000 molecules and do the equivalent of 2.3 million hours of science (counting the compute time for each core separately). The cluster ran only last week, so it's too early to find out what its impact on solar science will be. Still, from a computing standpoint, it's impressive.

Top 500 List (June, 2024)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899

https://www.top500.org/lists/top500/2024/06/

National Strategic Computing Initiative

- **July 29, 2015: President Obama issued an executive order**

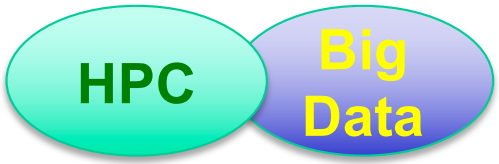
EXECUTIVE ORDER

- - - - - - -

CREATING A NATIONAL STRATEGIC COMPUTING INITIATIVE

BARACK OBAMA

By the authority vested in me as President by the Constitution and the laws of the United States of America, and to maximize benefits of high-performance computing (HPC) research, development, and deployment, it is hereby ordered as follows:



- **NSCI merges exaflop/s (10^{18} floating-point operations per second) high performance computing (HPC) & exabyte (10^{18} bytes) "big data" to advance the frontier of sciences, economic growth & national security**

cf. Cloud-native supercomputing