High-End Computing Trends

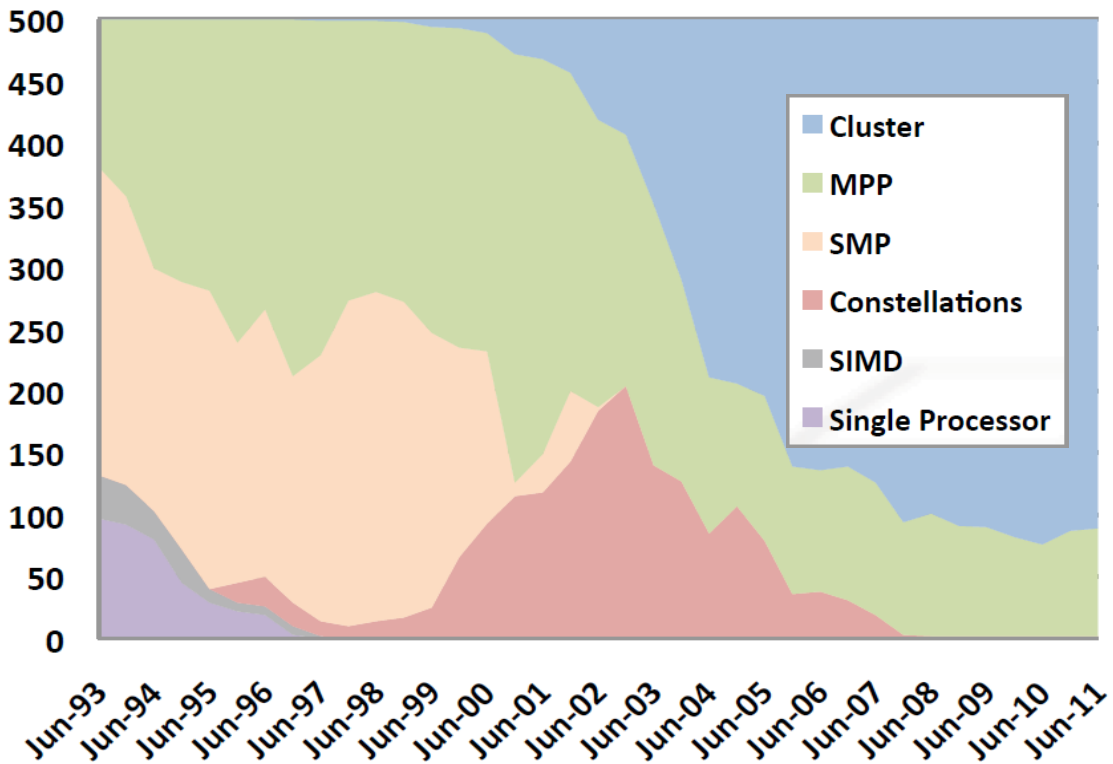
Speed → Scalability → Efficiency

Zhiwei Xu
Institute of Computing Technology (ICT)
Chinese Academy of Sciences

http://novel.ict.ac.cn/zxu/
zxu@ict.ac.cn

High-End Computing Pain Points

- Foresee future research trends
  - **architecture, execution model, programming model**

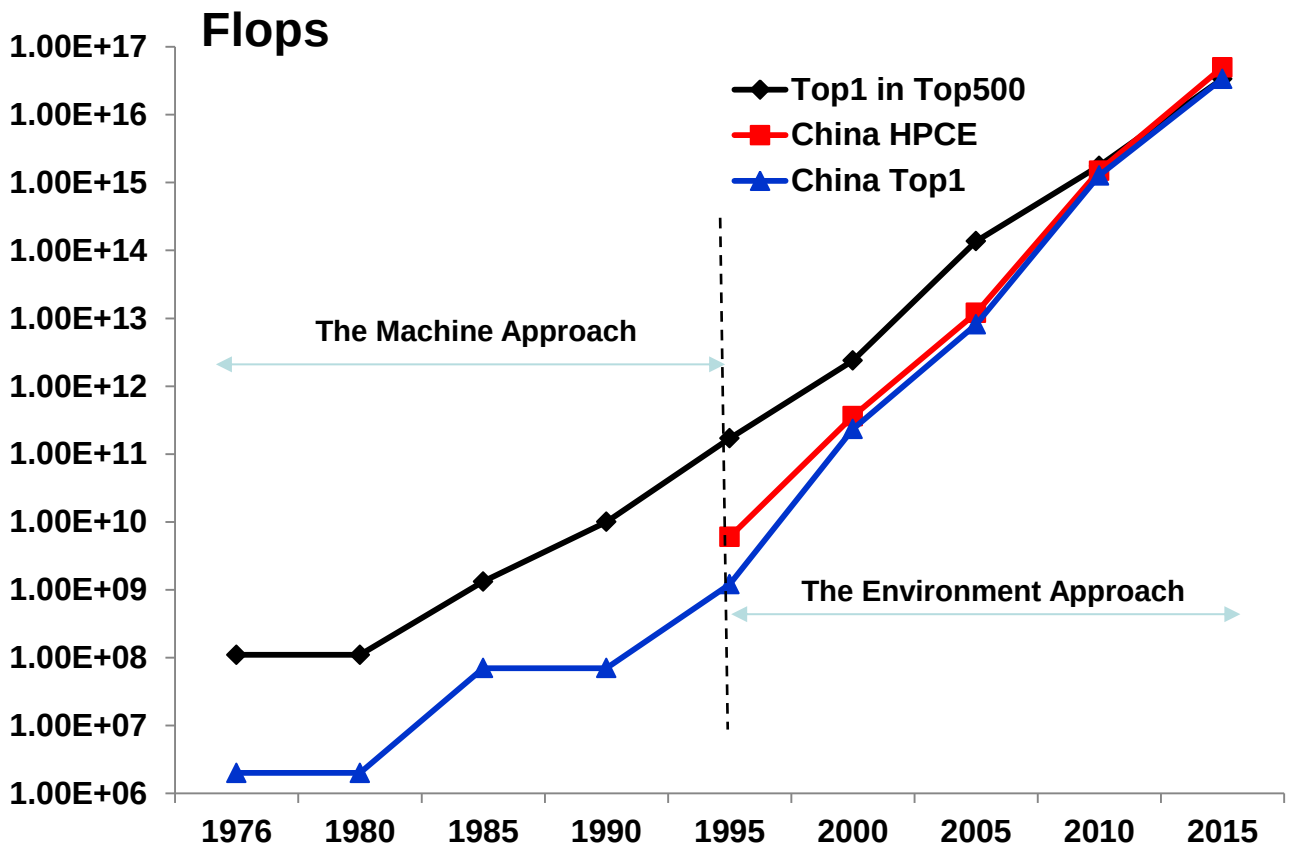


??

1976 **2015** **2020** **2025**

Look at the Past to See the Future

- Ecosystem
  - Stacks
  - Community
- Direction
  - **Goal**
  - Objectives
  - **Priority**
- 2035: ??



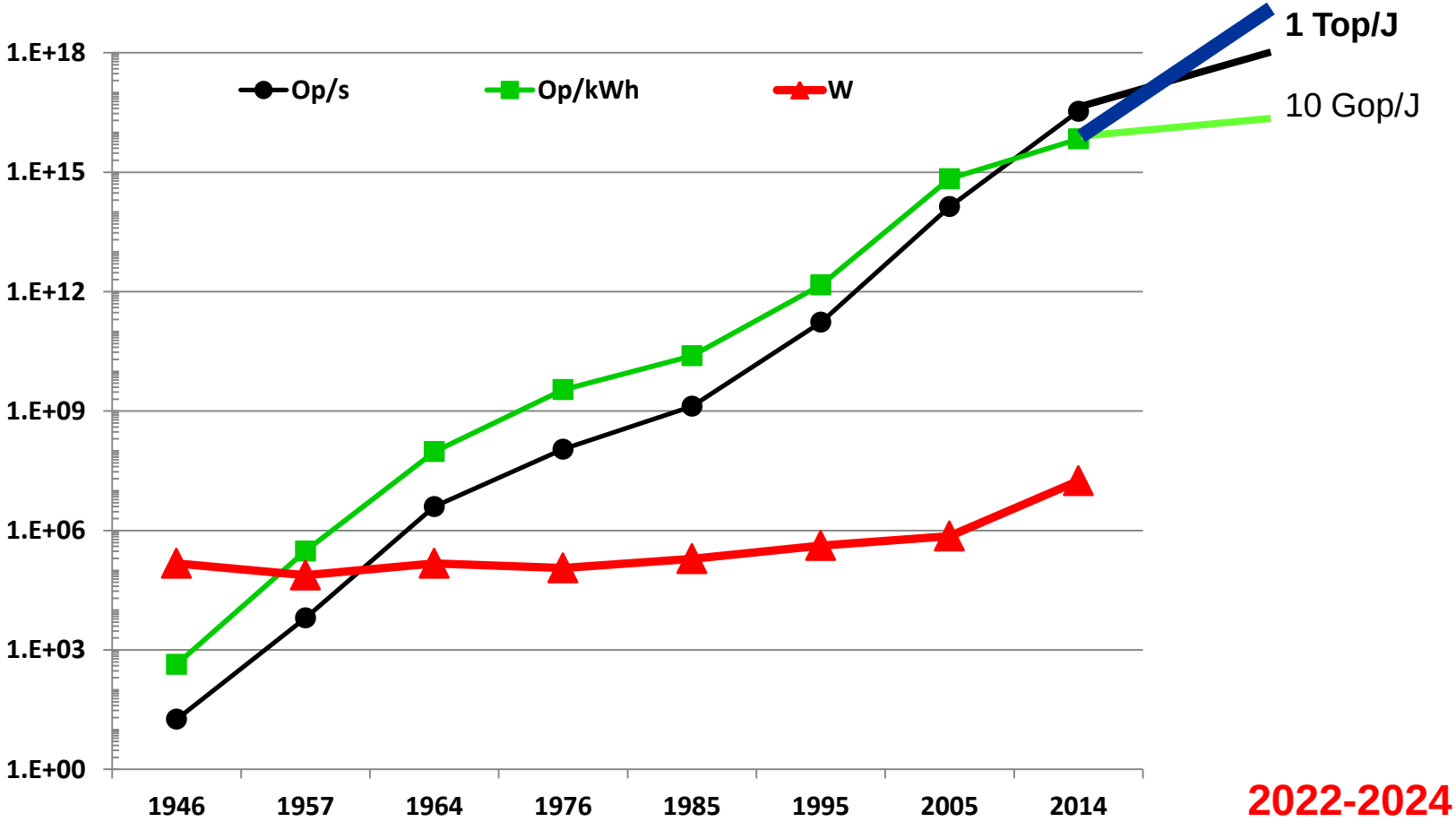
Growth trends of top HPC systems in China and in the World: before and after 1995

Flops Growth	1976 to 1995		1995 to 2015	
	Speed Increase	Annual Growth Rate	Speed Increase	Annual Growth Rate
China	600 times	40%	28 million times	136%
World	1550 times	47%	0.2 million times	84%

Fundamental Challenge

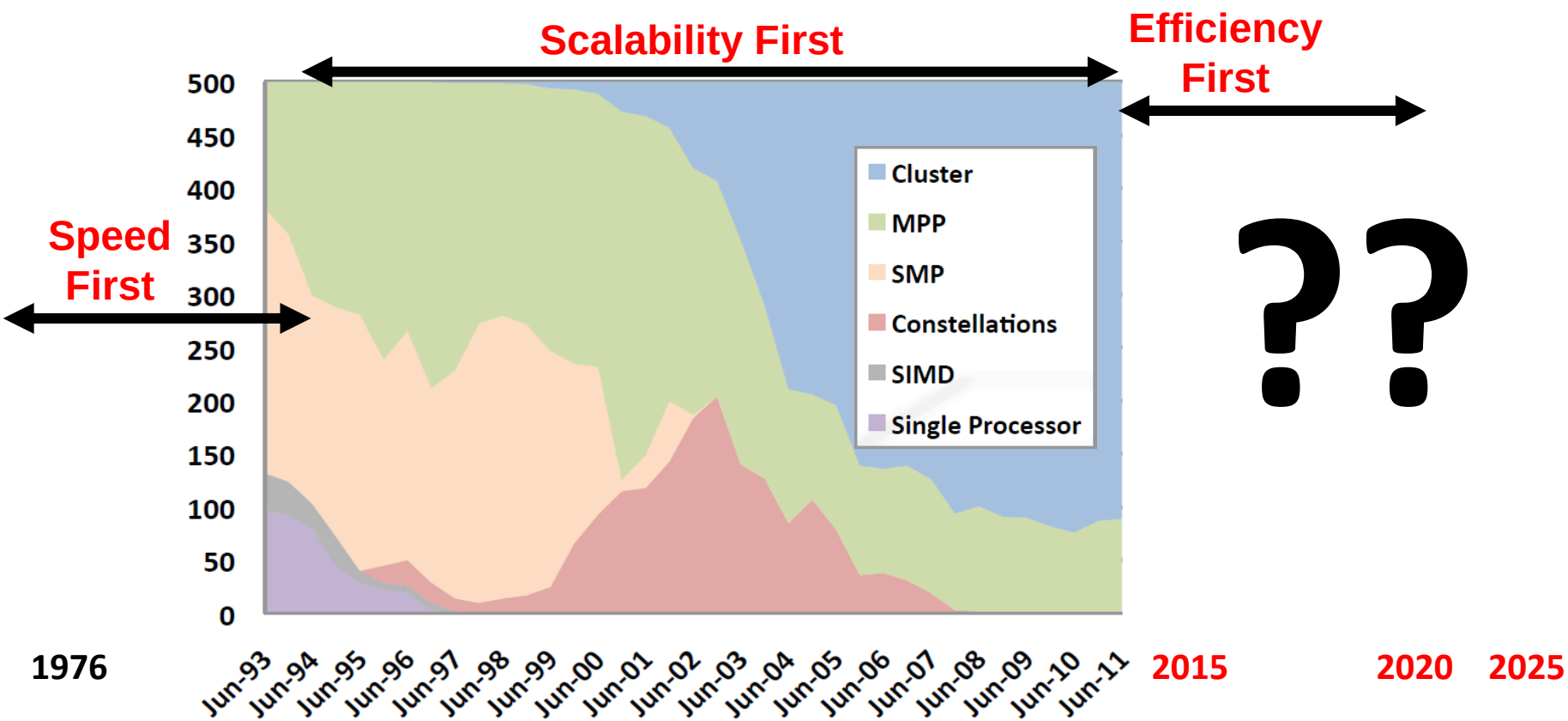
First time in 70 years

- Energy efficiency improvement lags behind speed growth → research goal: **1000 Gop/J**



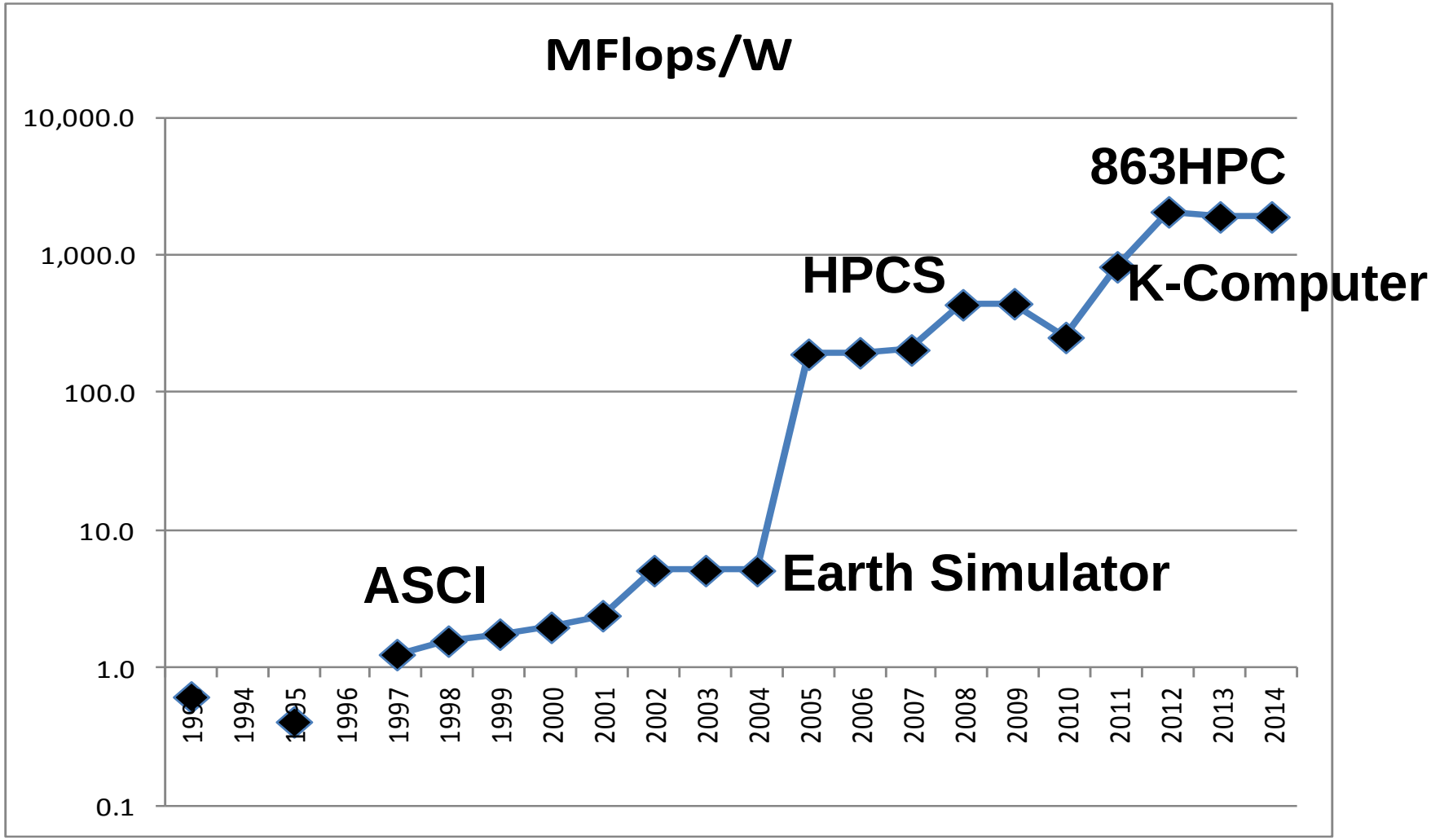
High-End Computing: Three Phases

- First priority went through two phases
  - Speed (flops), aka performance
  - Scalability: market scalability, problem scalability



Most Important Efficiency Metric

- Energy Efficiency: GOPS/W≈GOPJ



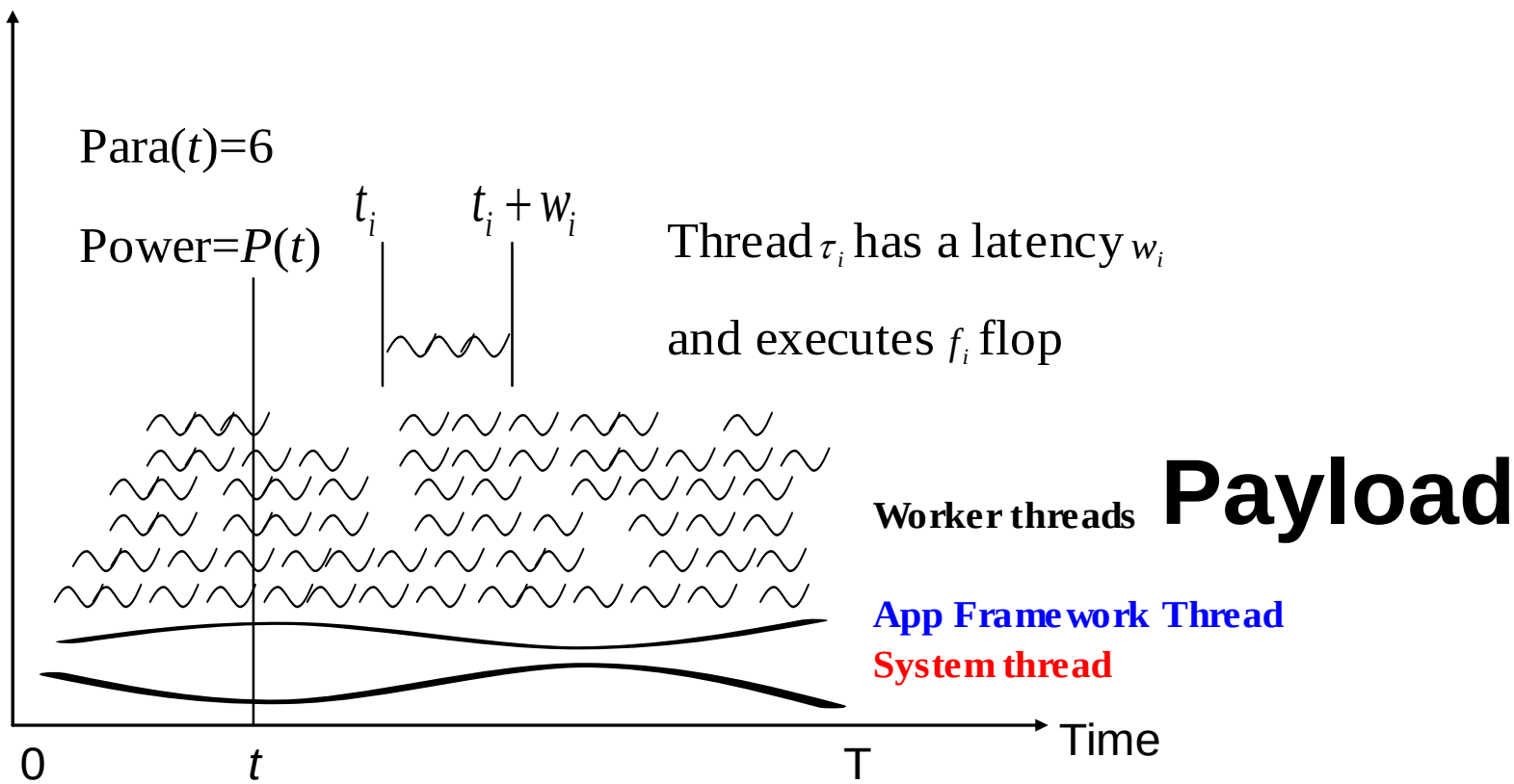
Progressive vs. Aggressive Approaches

- Progressive approaches: 1000 GOPS/W by 2035
- Aggresive approaches: 1000 GOPS/W by 2025
  - Model: equations relating speed, parallelism, power, energy
  - Technology: Makimoto's Wave
  - Architecture: Elastic Processor
  - Software:
    - "smart" libraries
    - application frameworks

	2010	2013	2022
Top500	DoE Jaguar 1.76 POPS@6.75MW **0.253 GOPS/W**	**1.9 GOPS/W**	1000 POPS@20MW **50 GOPS/W**
Green500	Dawning Nebula 1.27 POPS@2.58MW 0.492 GOPS/W	3.13 GOPS/W	?
Cloud-Sea Server	N/A	4 GOPS/W	System: **200 GOPS/W** Processor: **1000 GOPS/W**

Extending Little's Law to Characterize Energy Efficiency

- Focus on "threads per second" as a proxy of the performance objectives
  - Subject to latency, power, energy constraints
  - A thread is **a schedulable sequence of instruction executions with its own program counter**
    - POSIX thread, HW thread, Java thread, CUDA thread of GPU, Hadoop task, etc.
    - "Threads per second" serves as the neck of the performance metrics hourglass



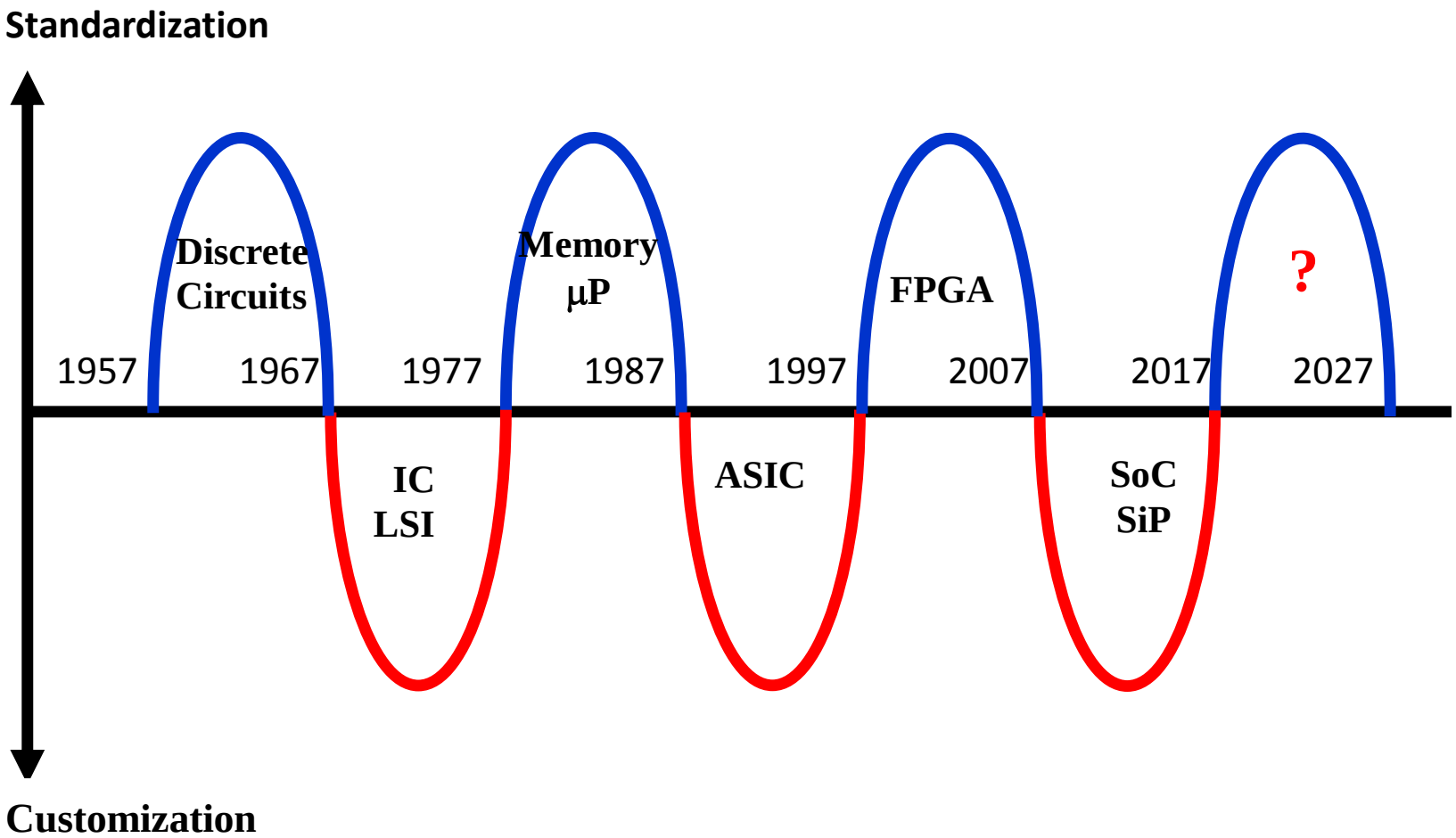
Assumptions and Observations

- Assume N threads $\{\tau 1,\ldots,\tau N\}$ are executed in a computer system in time period $[0, T]$, where
  - power and energy are additive; inactive threads consume no power
- Definitions of some average quantities
  - **Throughput** λ: threads per second, averaged over $[0, T]$
  - **Parallelism** L: number of active threads, averaged over $[0, T]$
  - **Latency** W: latency of a thread, averaged over $\{\tau 1,\ldots,\tau N\}$
  - **Power** P: Watts consumed by the system, averaged over $[0, T]$
  - **Energy** E: Joules consumed by a thread, averaged over $\{\tau 1,\ldots,\tau N\}$
  - **Work** F: Payload operations per thread (e.g., flop per thread)
  - **Speed** S: Payload operations per second
- Observations
  - Little's law: $\lambda = L / W$
  - New observations
    - $\lambda = P / E$
    - $\lambda = L \times (E/W) \times (1/E)$ **Throughput = Parallelism × Watts per thread × Threads per Joule**
    - $S = \lambda \times F = L \times F \times (E/W) \times (1/E) = L \times (E/W) \times (F/E)$
    - **Exaflops = 1 billion x 1 billion x (<20mW per thread) x (>1000 threads per Joule)**

Zhiwei Xu: Measuring Green IT in Society. IEEE Computer 45(5): 83-85 (2012)

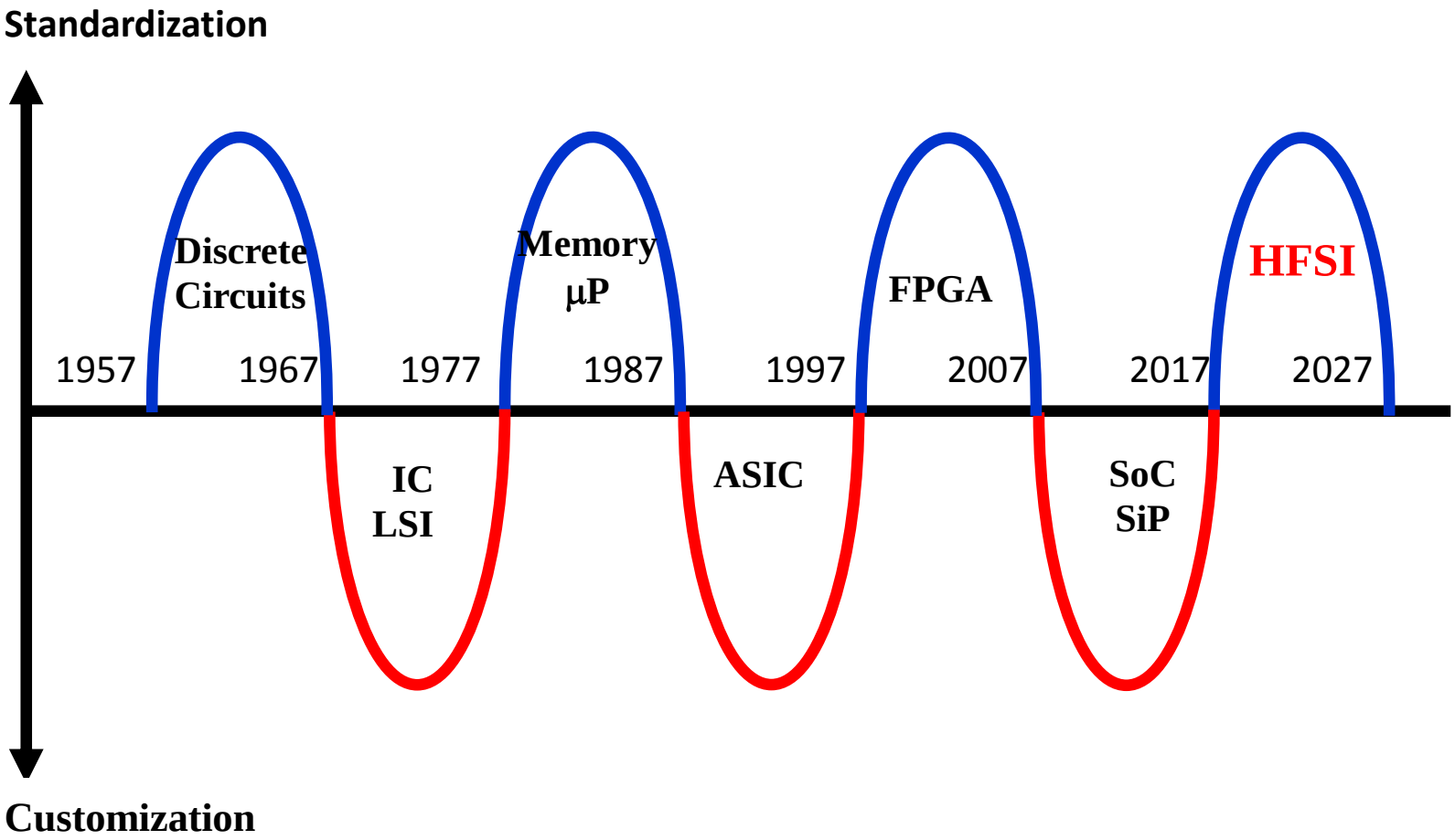
Makimoto's Wave

- Semiconductor technology will soon enter another phase change. But what is it?



Makimoto's Wave

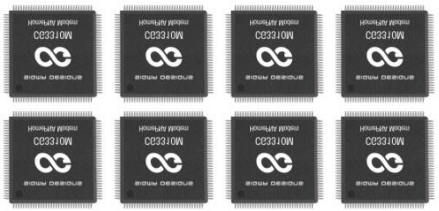
- HFSI: Highly Flexible Super Integration
- Redundant circuits can be shut off when not in use



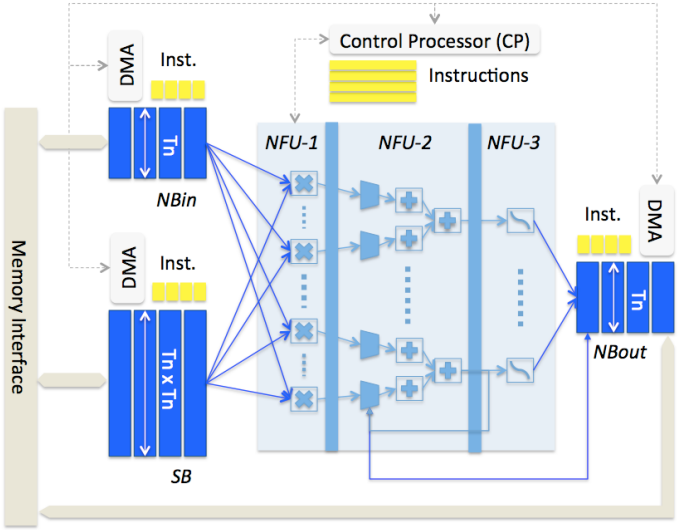
Elastic Processor

- A new architecture style (FISC)
  - Featuring **function instructions** executed by **programmable ASIC** accelerators
  - Targeting 1000 GOPS/W = 1 Top/J
  - Needing **smart libraries**

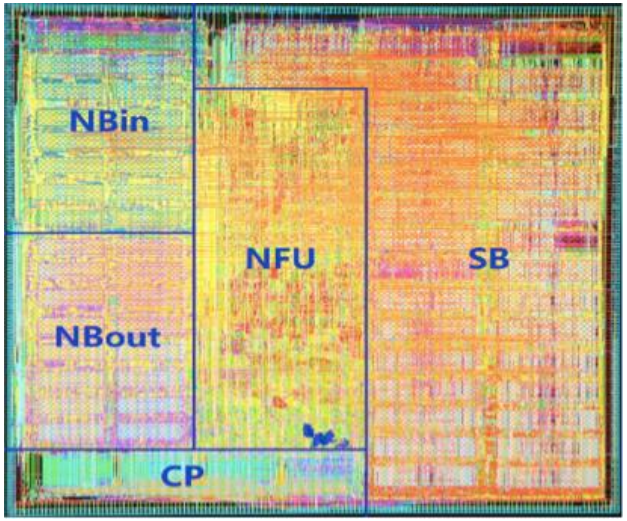
CISC Intel X86	RISC ARM	FISC Function Instruction Set Computer
Chip types: 10s	1K	10K
Power: 10~100W	1~10W	0.1~1W
Apps/chip: 10M	100K	10K



DianNao: A Neural Network Accelerator

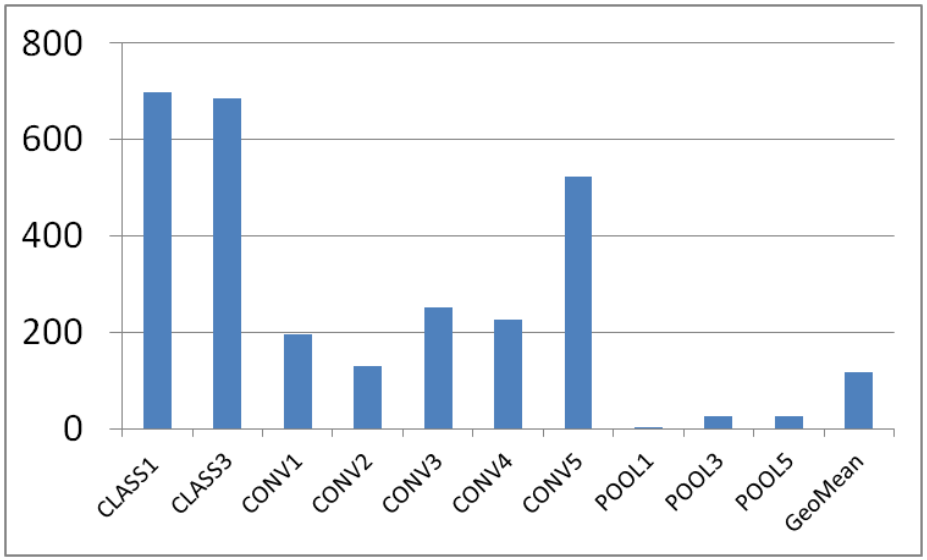


Architecture



IC Layout in GDSII

- Support multiple neural network algorithms, such as DNN, CNN, MLP,SOM
- Pre-tape-out simulation results:
  0.98GHz, 452 GOPS, 0.485W
  931.96 GOPS/W @ 65nm
- **ASPLOS 2014 Best Paper**

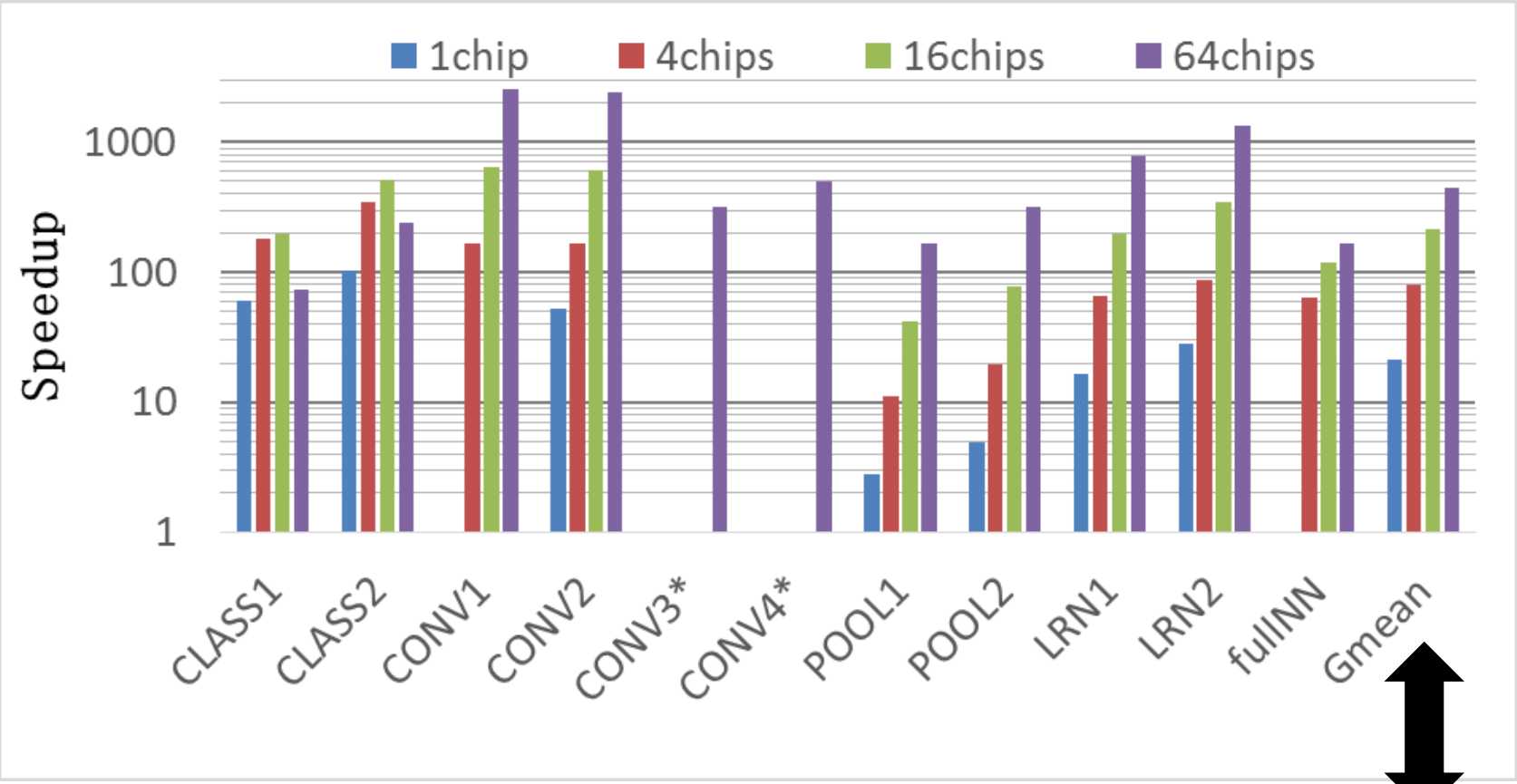


700 speedup over Intel Core i7

Three More Accelerators

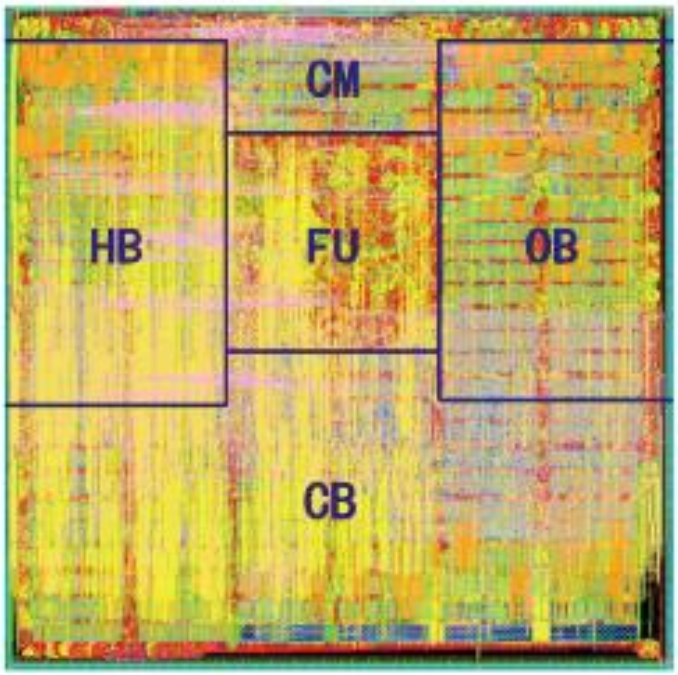
- **DaDianNao**: An NN supercomputer containing up to 64 chips
  - MICRO'14 best paper
  - **100-250 GOPS/W (@28nm)**
- **PuDianNao**: A polyvalent machine learning accelerator
  - ASPLOS'15
  - **300-1200 GOPS/W**
- **ShiDianNao**: A vision accelerator for embedded devices (cameras)
  - ISCA'15
  - **2000-4000 GOPS/W (16-bit)**
- Compared to **931 GOPS/W @65nm** for **DianNao**

DaDianNao: An NN Supercomputer



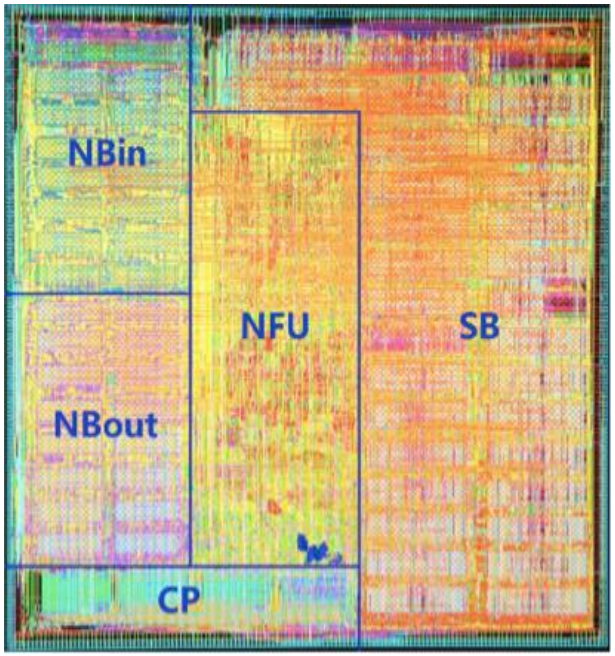
- In average, 450x speedup and 150x energy saving over K20 GPU for a 64-chip system

PuDianNao



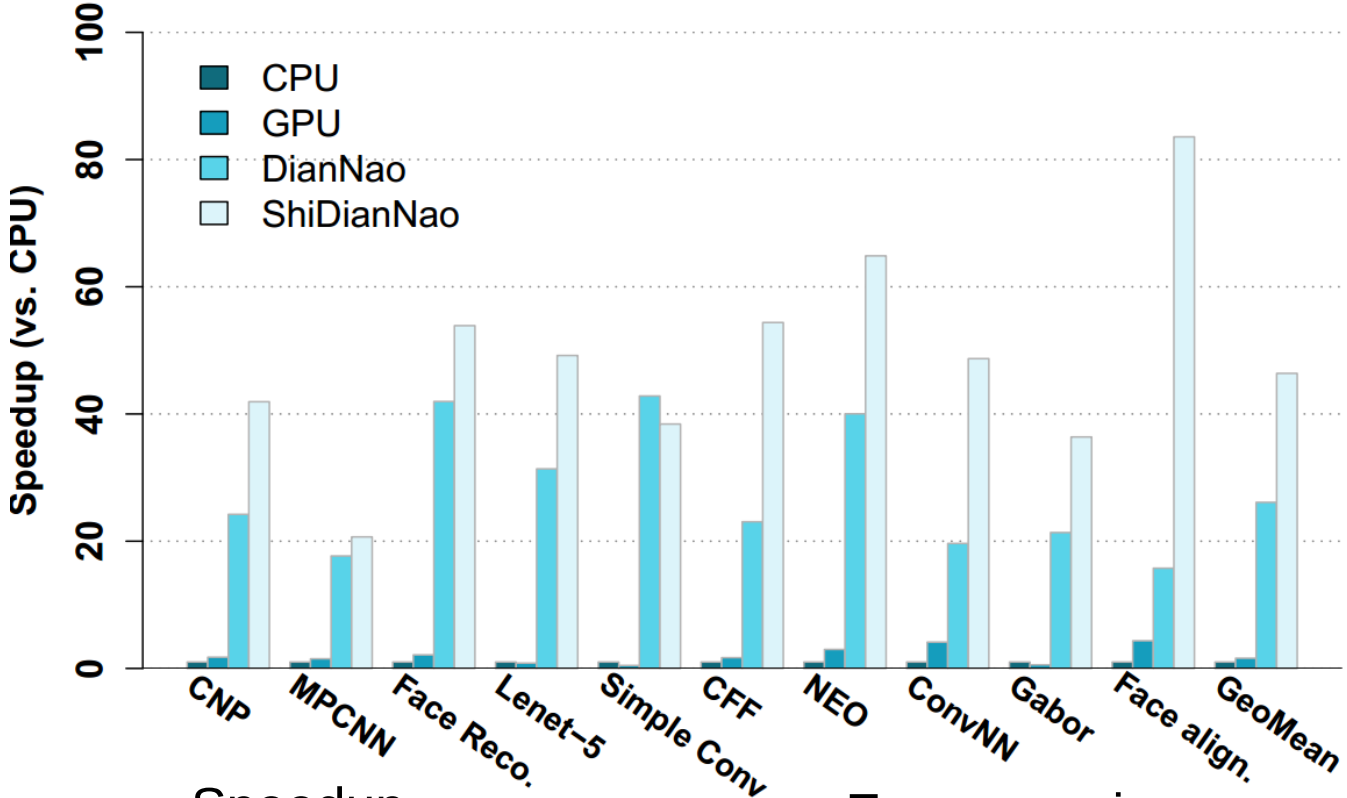
- Area: 3.51 mm^2
- Power: 596 mW
- Freq: 1 GHz
- Supporting a dozen types of ML algorithms: CNN/DNN, LR, Kmeans, SVM, NB, KNN, CT, …

DianNao



- Area: 3.02 mm2
- Power: 485 mW
- Freq: 0.98 GHz
- Supporting CNN/DNN

ShiDianNao: An Vision Accelerator for Embedded Devices



Speedup

46.38x vs. CPU

28.94x vs. GPU

1.87x vs. DianNao

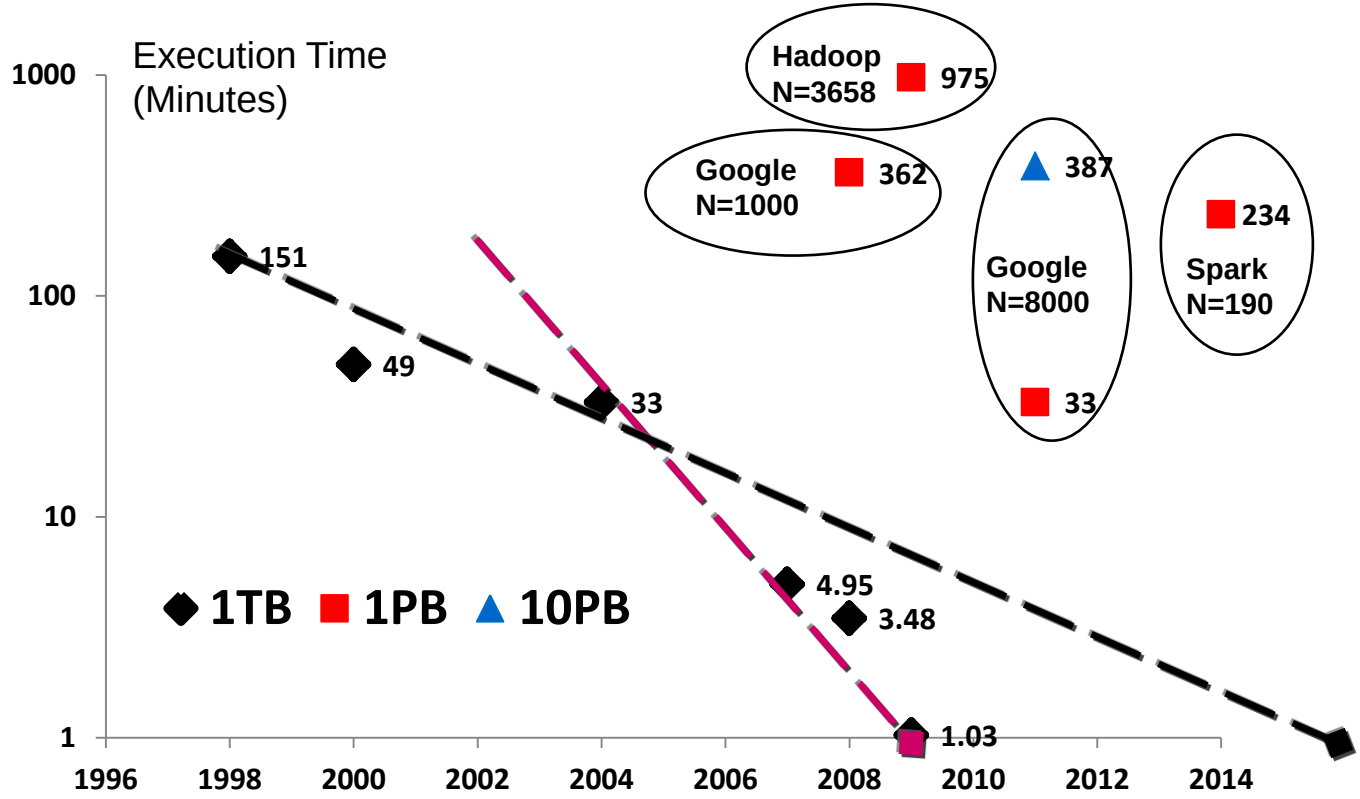
Energy saving

4688.13x vs. GPU

63.48x vs. DianNao

Jim Gray's Data Challenge: Four Phases

1. Terasort challenge raised (sort 1TB in 1 minute)
2. Speed growth: TB/minute in 2009 (ad hoc methods)
3. Data size growth: TB→PB→10PB (Map-Reduce)
4. Efficiency: 1PB sorted on 190 AWS instances in 2014



?

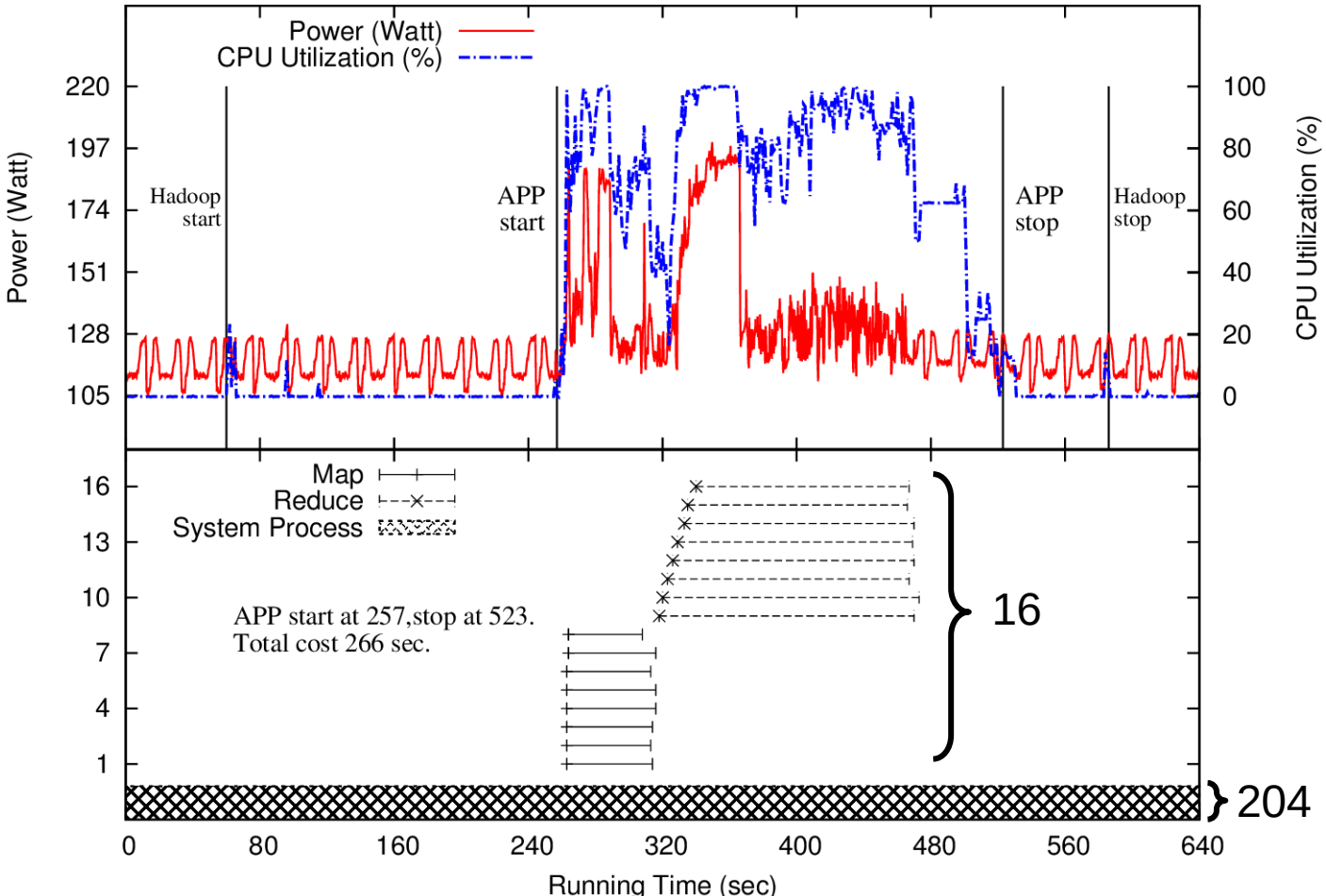
Efficiency becomes top priority

?

2020

Hadoop Efficiency Is Low

- Lacks a high-performance communication substrate
  - Use HTTP, RPC, direct Sockets over TCP/IP to communicate
  - Can MPI be used for big data?



Speed Efficiency

(Sustained/Peak)

Payload:	**0.002%**
Linpack:	**94.5%**
Total op:	4.22%
Instruction:	4.72%

Energy Efficiency

(Operations per Joule)

Payload:	**1.55×10^4**
Linpack:	**7.26×10^8**
Total op:	2.20×10^7
Instruction:	2.45×10^7

Direct MPI Use not Easy or Scalable

```
//MapReduce
map (String lineno, String
contents) {
        for each word w in contents
{
                EmitIntermediate(w, 1);
    }
}

reduc                    ) {
    incre                  a    );
}
```

LOC : 110

```
//MPI
process mapper:
1st> load input
2nd> parse token
3rd> MPI_Send (serialization)
…

proces                  d
1st
(Dese
2nd> inc
3rd> save ou    out
…
```

LOC: 850

WordCount via MapReduce：Scalable over 1GB, 1TB, 1PB …

Desired Sort Code via DataMPI: Scalable and Easy to Write

```
1:  public class Sort {
2:      public static void main(String[] args) {
3:          try {
4:              int rank, size;
5:              Map<String, String> conf = new HashMap<String, String>();
6:              conf.put(MPI_D_Constants.KEY_TYPE, java.lang.String.class.getName());
7:              conf.put(MPI_D_Constants.VALUE_TYPE, java.lang.String.class.getName());
8:              MPI_D.Init(args, MPI_D.Mode.Common, conf);
9:              if (MPI_D.COMM_BIPARTITE_O != null) {
10:                 rank = MPI_D.Comm_rank(MPI_D.COMM_BIPARTITE_O);
11:                 size = MPI_D.Comm_size(MPI_D.COMM_BIPARTITE_O);
12:                 String[] keys = loadKeys(rank, size);
13:                 if (keys != null) {
14:                     for (int i = 0; i < keys.length; i++) {
15:                         MPI_D.Send(keys[i], "");
16:                     }
17:                 }
18:             } else {
19:                 rank = MPI_D.Comm_rank(MPI_D.COMM_BIPARTITE_A);
20:                 size = MPI_D.Comm_size(MPI_D.COMM_BIPARTITE_A);
21:                 Object[] keyValue = MPI_D.Recv();
22:                 while (keyValue != null) {
23:                     System.out.println("Task " + rank + " of " + size + " key is "
24:                         + ((String) keyValue[0]) + ", value is " + ((String) keyValue[1]));
25:                     keyValue = MPI_D.Recv();
26:                 }
27:             }
28:             MPI_D.Finalize();
29:         } catch (MPI_D_Exception e) {
30:             e.printStackTrace();
31:         }
32:     }
33: }
```

init

rank/size

send

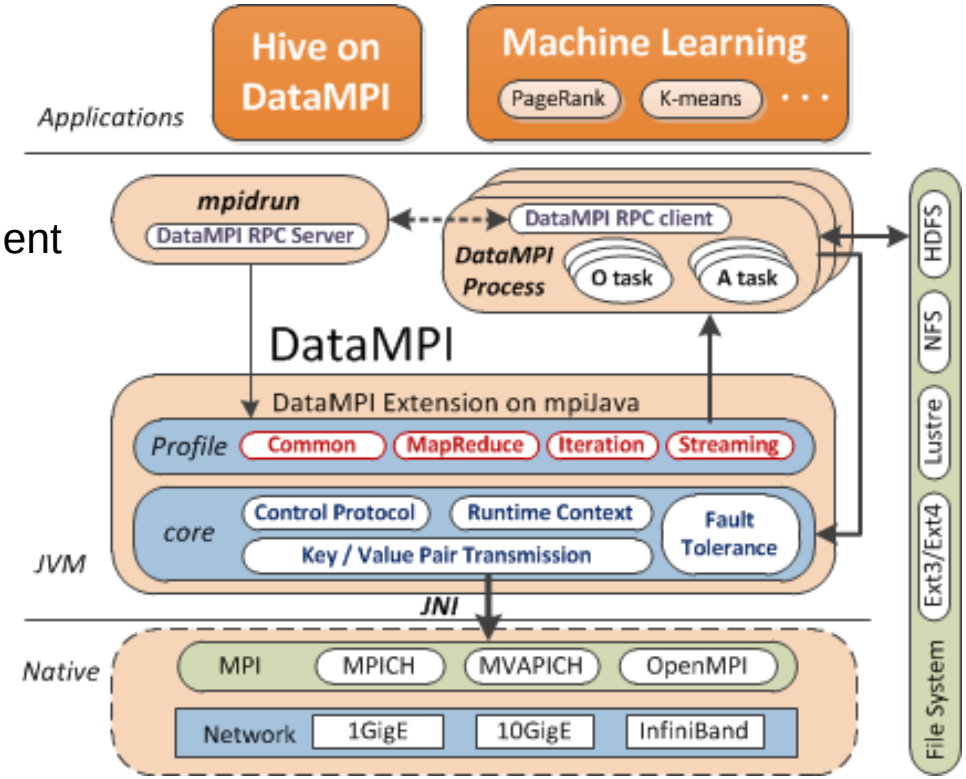
recv

finalize

33 lines of code
1 GB, 1 TB, 1PB

DataMPI.ORG

- **Core**
  - Execution pipeline
  - Key-value communication
  - Native direct IO for buffer management
  - Key-value based checkpoint
- **Profiles**
  - Additional code sets
  - Each profile for a typical mode
- **mpidrun**
  - Communicator creation
  - Dynamic process management
  - Data-centric task scheduling



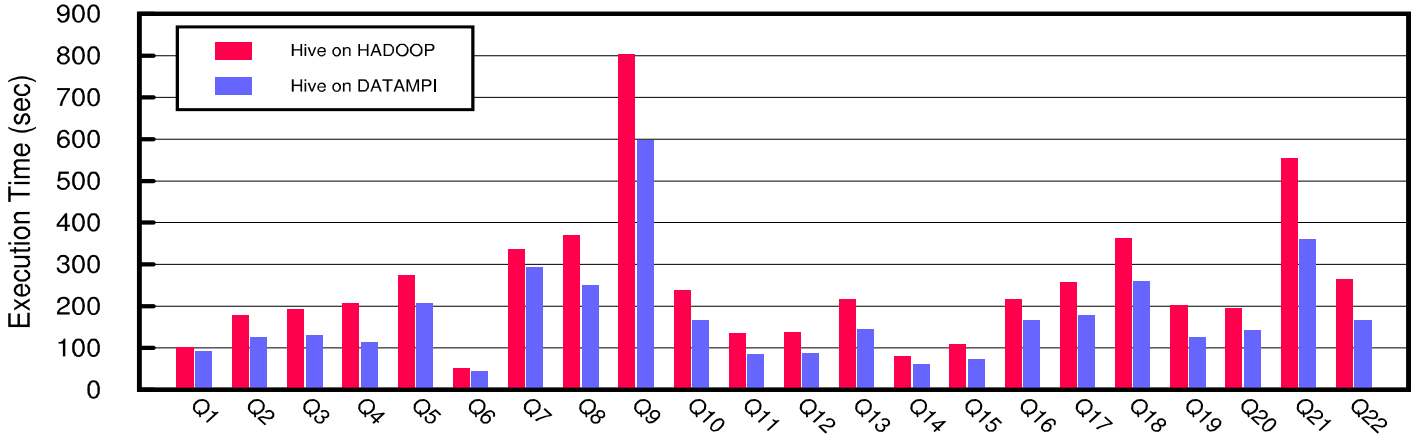
Command: ***$ mpidrun -f <hostfile> -O <n> -A <m> -M <mode> -jar <jarname> <classname> <params>***

Example: ***$ mpidrun -f ./conf/hostfile -O 64 -A 64 -M COMMON -jar ./benchmark/dmb-benchmark.jar dmb.Sort /text/64GB-text /text/64GB-output***

Hive on DataMPI

A first attempt to propose a general design for fully supporting and accelerating data warehouse systems with MPI

- **Functionality & Productivity & Performance**
  - Support Intel **HiBench** (2 micro benchmark queries) & **TPC-H** (22 app queries)
  - Only **0.3K** LoC modified in Hive
  - HiBench: **30%** performance improvement on average
  - TPC-H: **32%** improvement on average, up to **53%**



Performance Benefits with 40 GB data for 22 TPC-H queries

Lu Chao, Chundian Li, Fan Liang, Xiaoyi Lu, Zhiwei Xu. *Accelerating Apache Hive with MPI for Data Warehouse Systems*. ICDCS 2015, Columbus, Ohio, USA, 2015

References

- Elastic Processor
  - Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam, "ShiDianNao: Shifting Vision Processing Closer to the Sensor", ISCA'15.
  - Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Temam, Xiaobing Feng, Xuehai Zhou, and Yunji Chen, "PuDianNao: A Polyvalent Machine Learning Accelerator", ASPLOS'15.
  - Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam, "DaDianNao: A Machine-Learning Supercomputer", MICRO'14.
  - Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam, "DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning", ASPLOS'14.
- DataMPI
  - Lu Chao, Chundian Li, Fan Liang, Xiaoyi Lu, Zhiwei Xu. Accelerating Apache Hive with MPI for Data Warehouse Systems. ICDCS 2015, Columbus, Ohio, USA, 2015
  - Xiaoyi Lu, Fan Liang, Bing Wang, Li Zha, Zhiwei Xu: DataMPI: Extending MPI to Hadoop-Like Big Data Computing. IPDPS 2014: 829-838
  - Xiaoyi Lu, Bing Wang, Li Zha, Zhiwei Xu: Can MPI Benefit Hadoop and MapReduce Applications? ICPP Workshops 2011: 371-379
- Energy Efficient Ternary Computing
  - Zhiwei Xu: High-Performance Techniques for Big Data Computing in Internet Services. Invited speech at SC12, SC Companion 2012: 1861-1895
  - Zhiwei Xu: Measuring Green IT in Society. IEEE Computer 45(5): 83-85 (2012)
  - Zhiwei Xu: How much power is needed for a billion-thread high-throughput server? Frontiers of Computer Science 6(4): 339-346 (2012)
  - Zhiwei Xu, Guojie Li: Computing for the masses. Commun. ACM 54(10): 129-137 (2011)
- **http://novel.ict.ac.cn/zxu/#PAPERS**

zxu@ict.ac.cn http://novel.ict.ac.cn/zxu/