Uri weiser

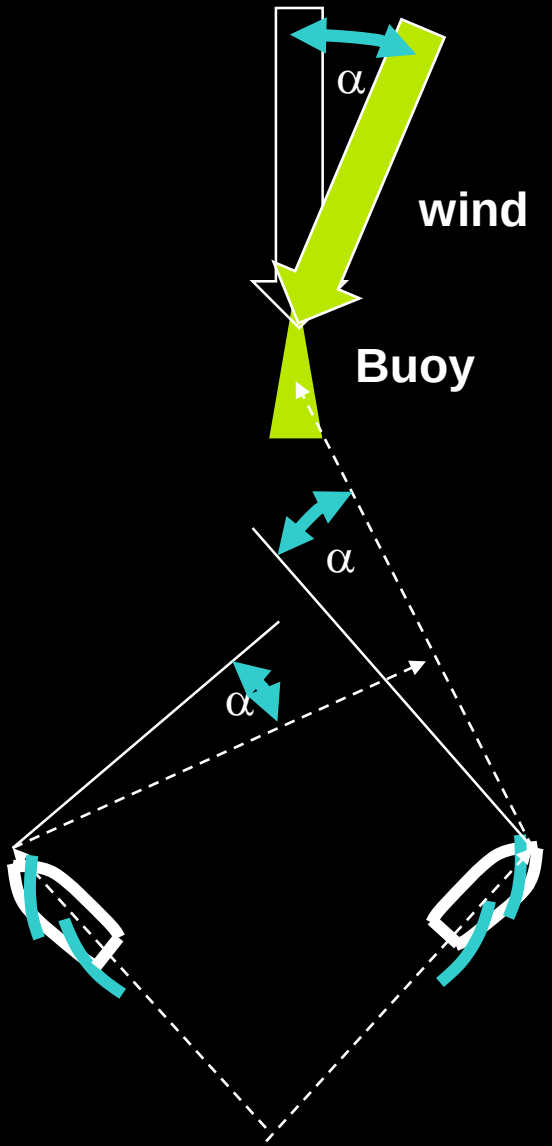
Assignment by Yale

- **“Your vision of the future of computer architecture. From the man who gave us MMX, refused to kill the golden goose, and worked for a time in the same box with Mark McDermott”**

Situation

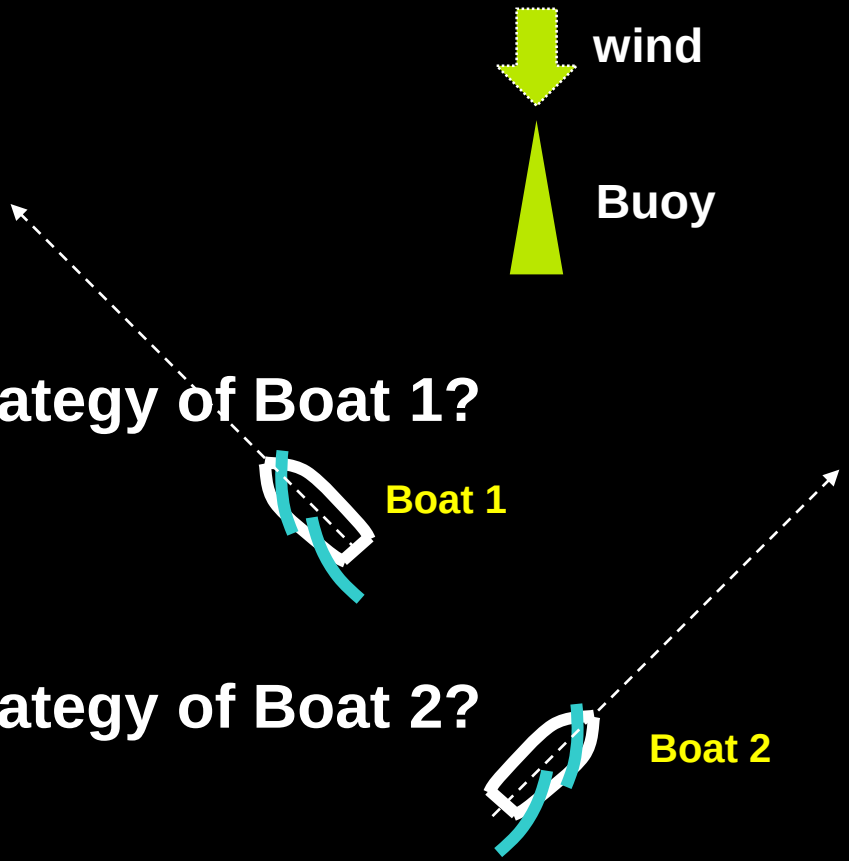
- Flew 11,482 km to greet Yale ☺
- Have to fight again with Bob
- What can I fill-in after this extraordinary speaker?
- Defiantly a challenge

My moto:
Sailing - wind shift



Sailing competition

getting first

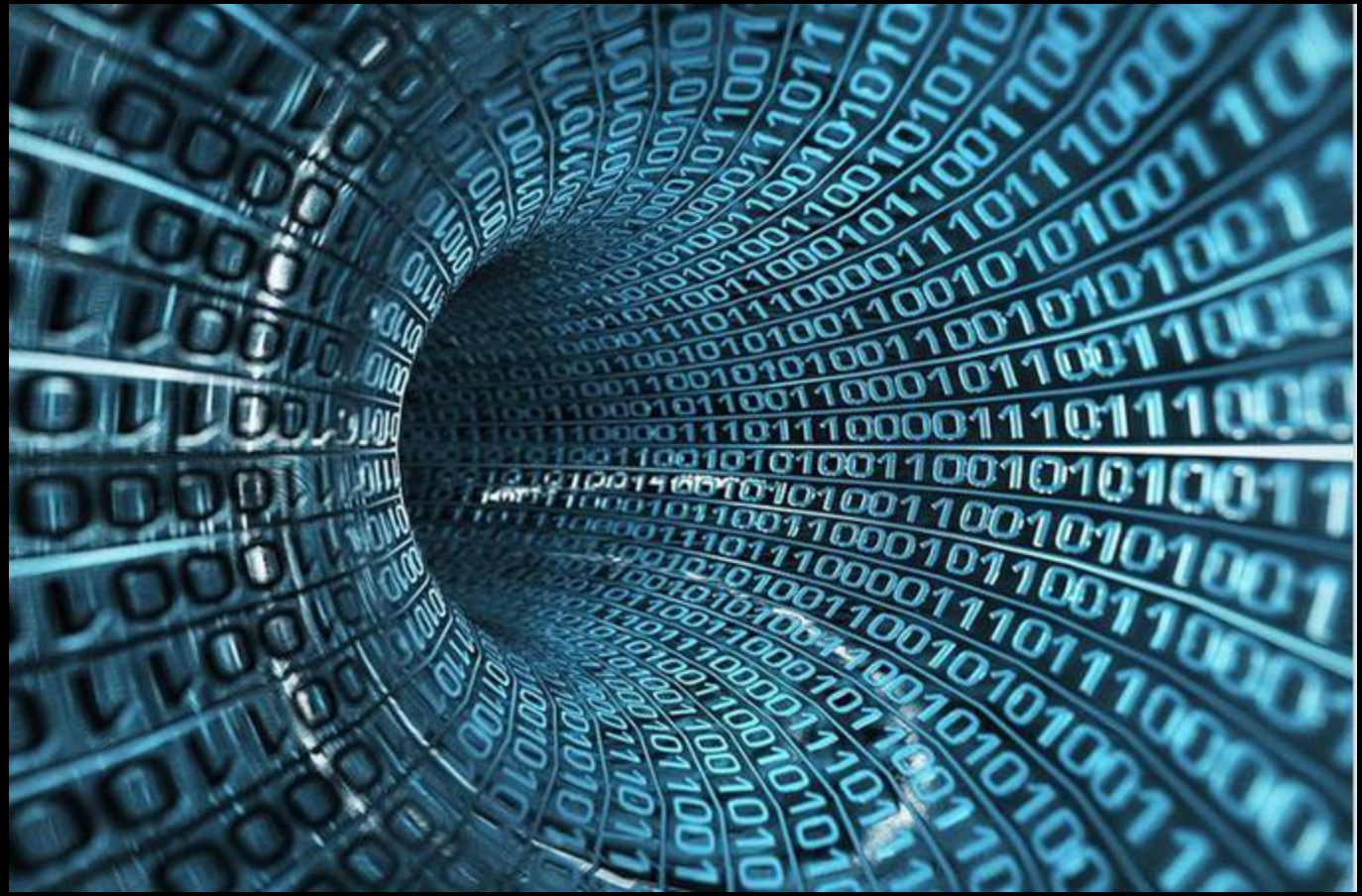


- What is the Strategy of Boat 1?

- What is the Strategy of Boat 2?

My Moto: Do not follow ➔ Invent

Uri Weiser
Professor
Technion
Haifa, Israel



Future Architecture Research
Big Data environment

Outline

- Big Data need ➔ reduction in energy/task
- Power/Energy - the opportunities
  - Heterogeneous systems – past thoughts
    - Resource allocation in a Heterogeneous system
  - Efficient computation ➔ reduction of Data movements
    - Avoid-the-Valley – past thoughts, deferent perspective
    - Big Data execution – where should we preform execution of “Funnel” functions

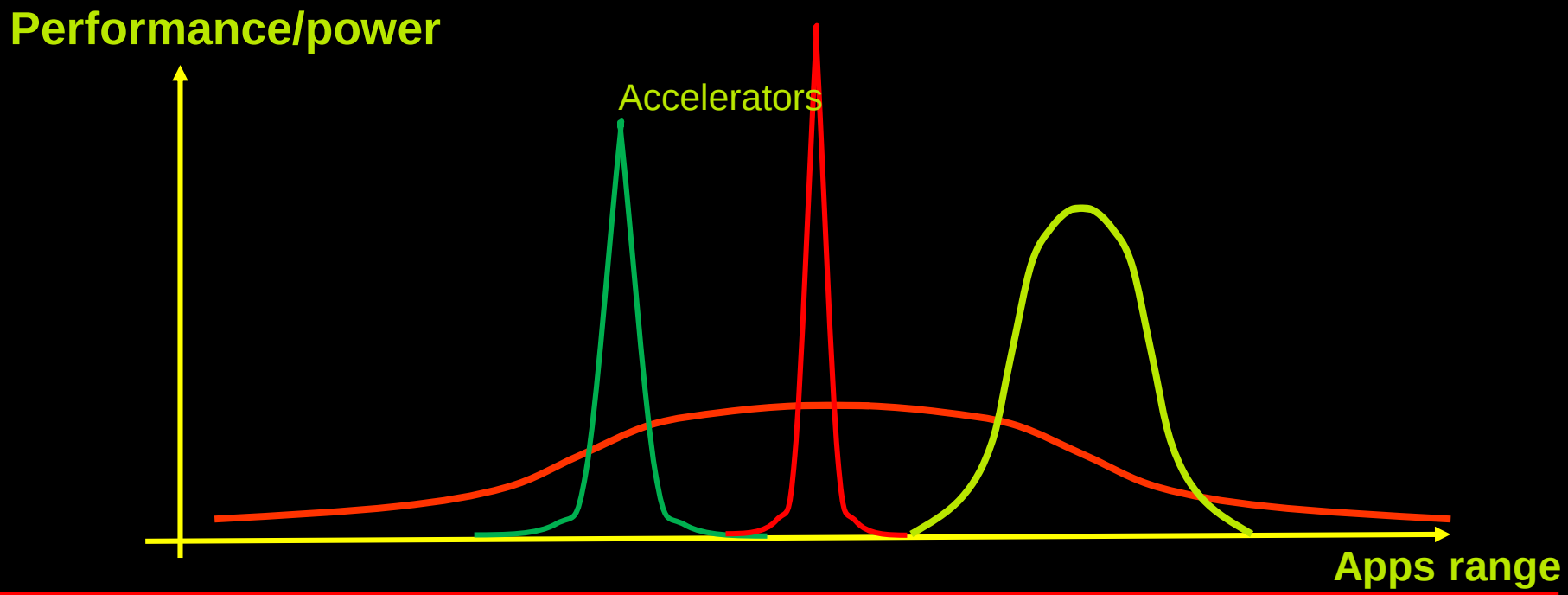
Big Data ➔ reduction in energy/task

- Hadoop/Spark Calls for multiple computing engines taking care of “ONE TASK”
- Computing Centers’ attention was shifted from Performance toward energy saving
- The need for huge amount of processing ➔ huge consumption of energy
- See Google centers…

Power/Energy the opportunities

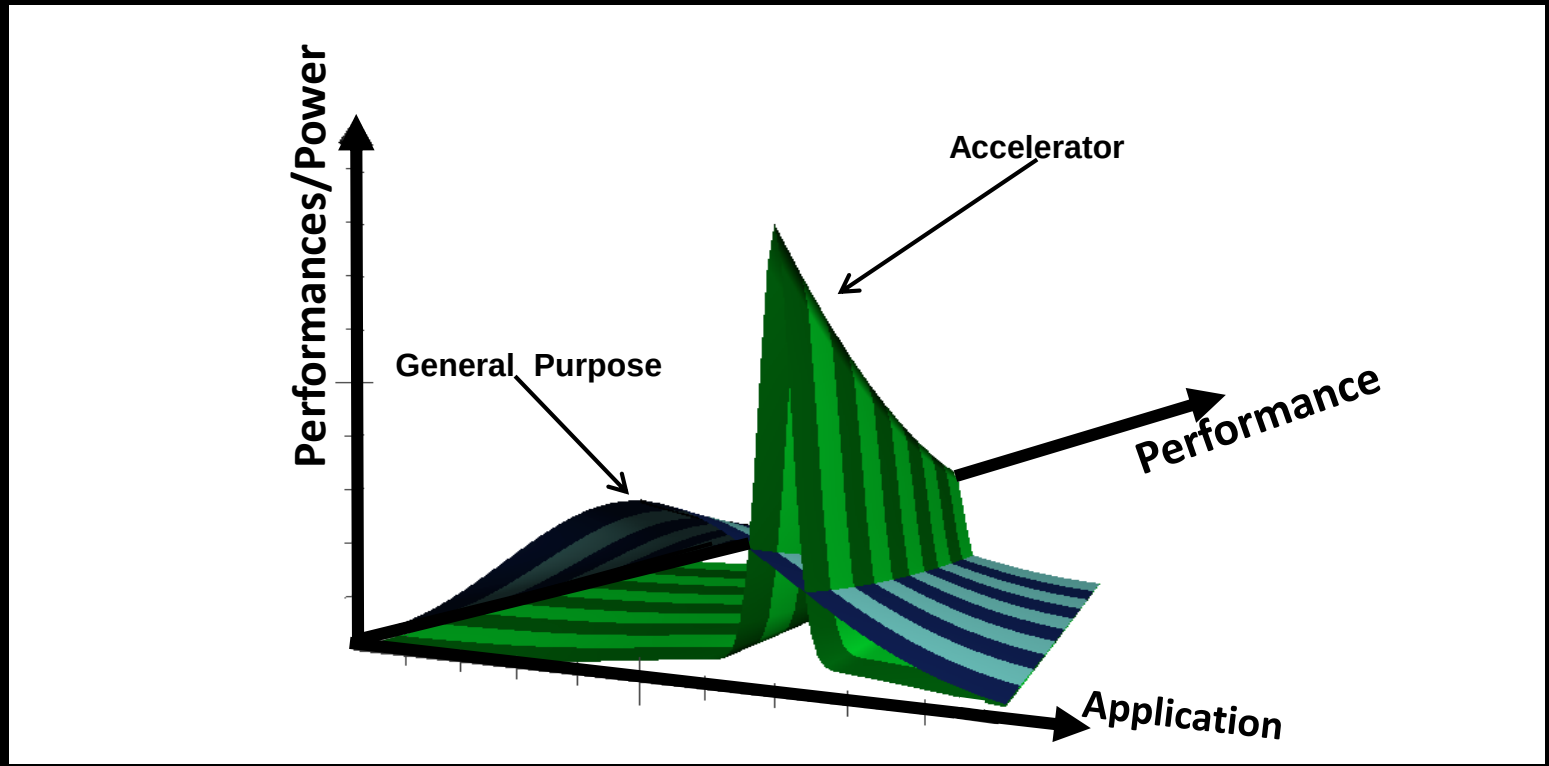
- Heterogeneous Systems –Past findings
  - Resource allocation in a Heterogeneous system - MA
- Efficient computation ➔ reduction of Data movements
  - Avoid-the-Valley – past thoughts, deferent perspective
  - Big Data execution – where should we preform execution of “Funnel” functions

Heterogeneous Computing: Application Specific Accelerators

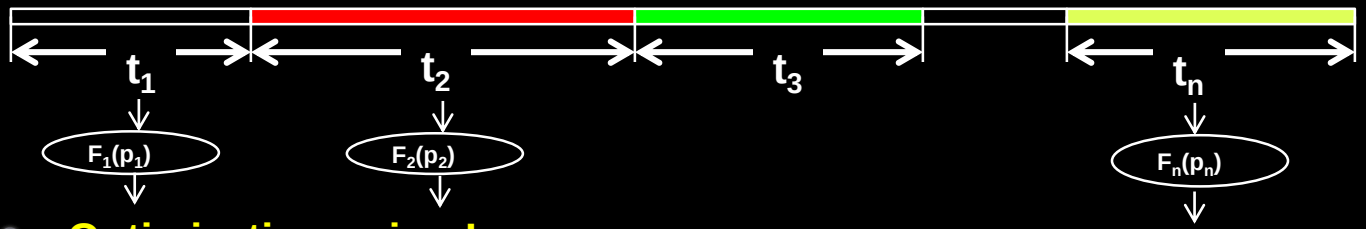


Continue performance trend using Heterogeneous computing to bypass power and energy hurdles

Heterogeneous Computing



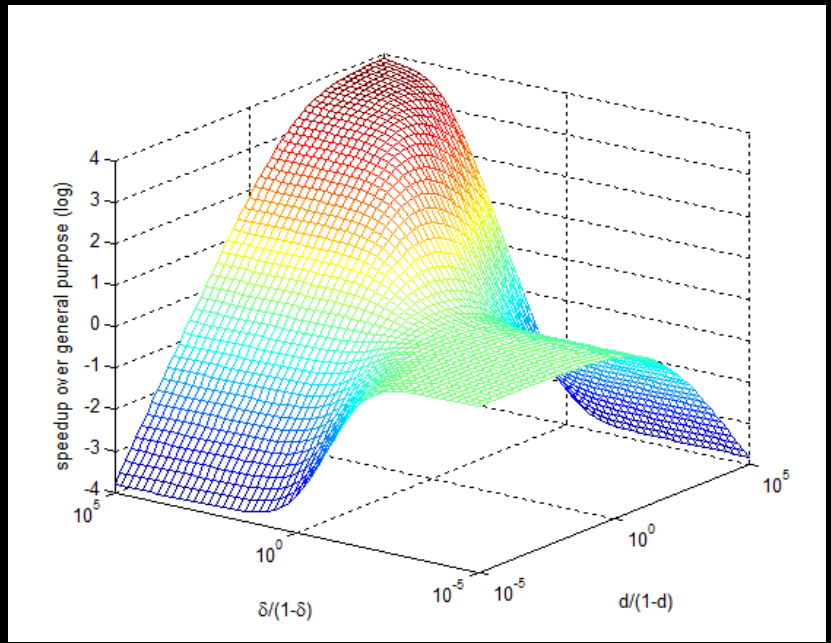
MultiAmdahl:



- **Optimization using Lagrange multipliers**
  Minimize execution time (T)
  under a Area (a) constraint

$$t_j F'_j(p_j) = t_i F'_i(p_i)$$

F'= derivation of the accelerator function
p_i = Power of the i-th accelerator
t_i = Execution time on reference computer



Power/Energy the opportunities

- Heterogeneous Systems –Past findings
  - Resource allocation in a Heterogeneous system
- Efficient computation ➔ reduction of Data movements
  - Avoid-the-Valley – past thoughts, deferent perspective
  - Big Data execution – where should we preform execution of “Funnel” functions

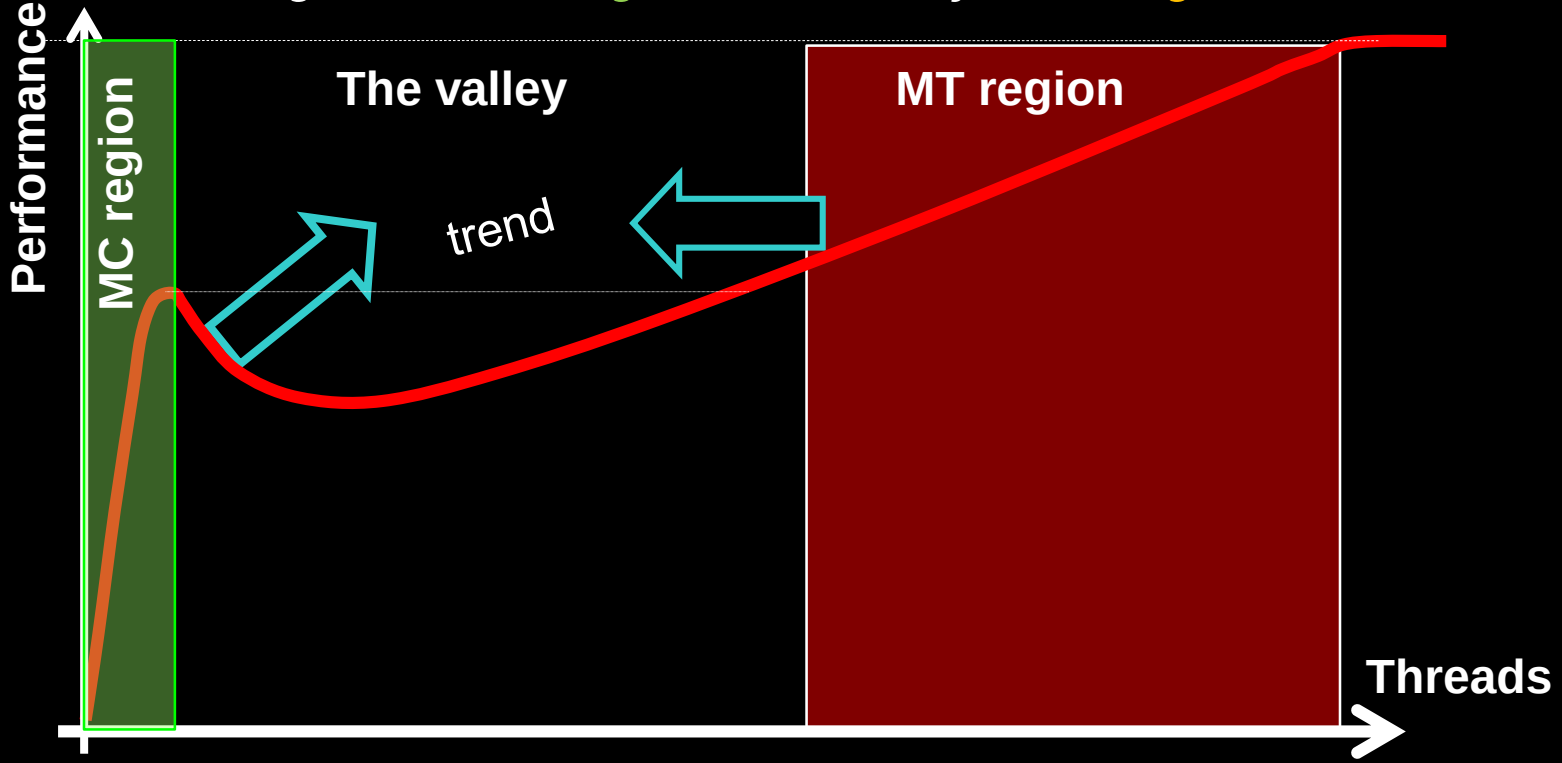
Power/Energy the opportunities

Efficient computation ➔ reduction of Data movements

- **Avoid-the-Valley – past research ➔ power implications**
- **The Funnel PreProcessing (FPP):**
  ak'a "In-Place-Computing" =
  Compute at the most energy effective place

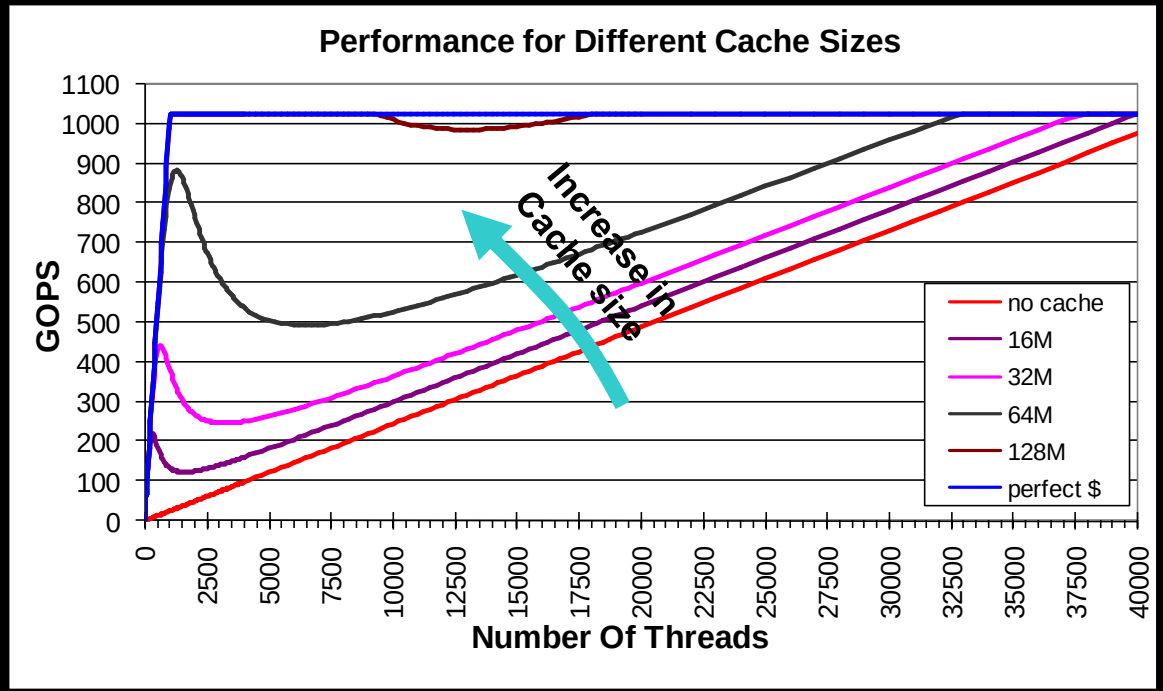
Avoid-the-valley: Many cores behind a common cache running many threads

- *Three regions: MC Region , the valley, MT Region*



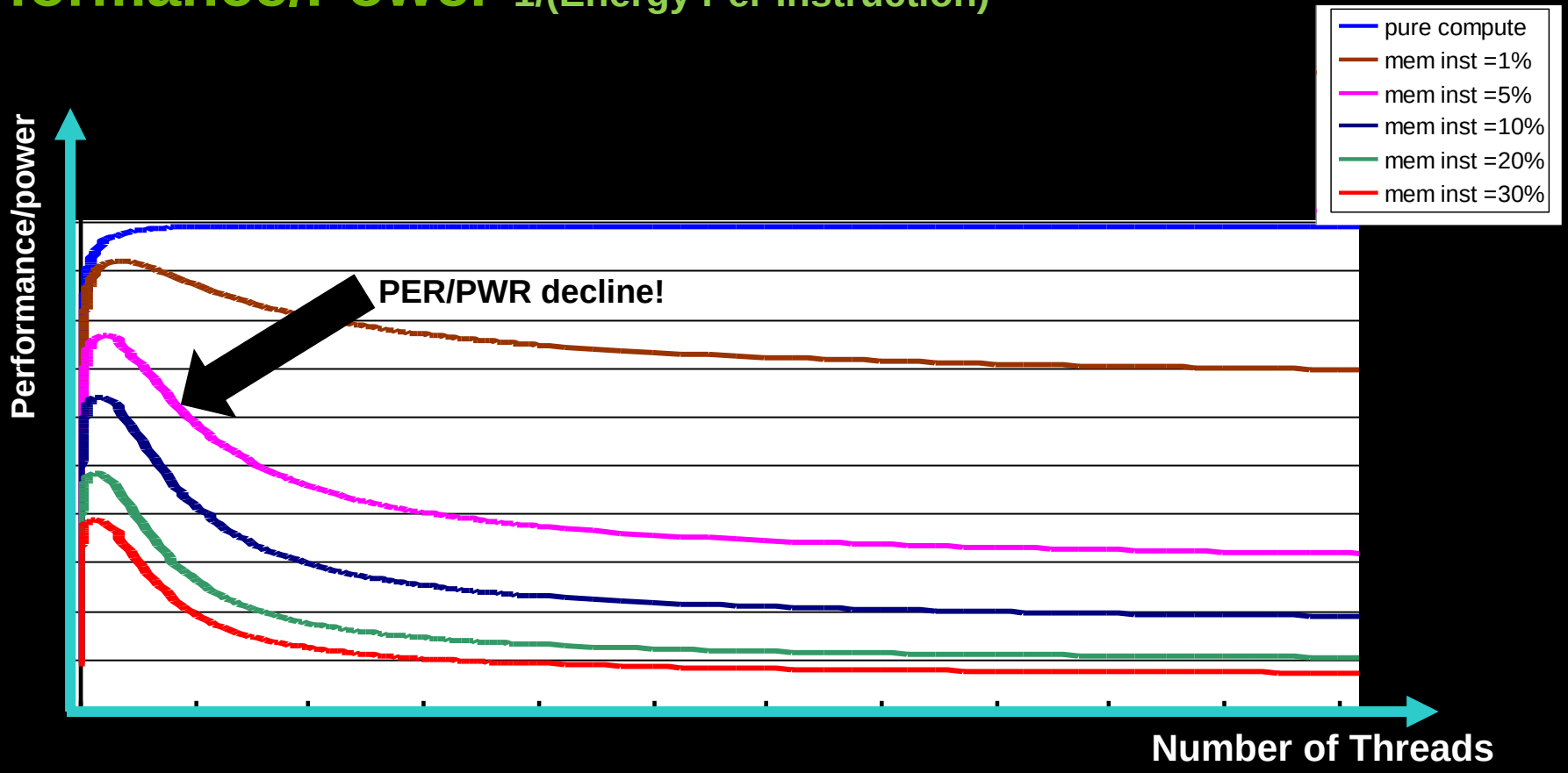
Avoid the Valley

Parameter: Cache Size

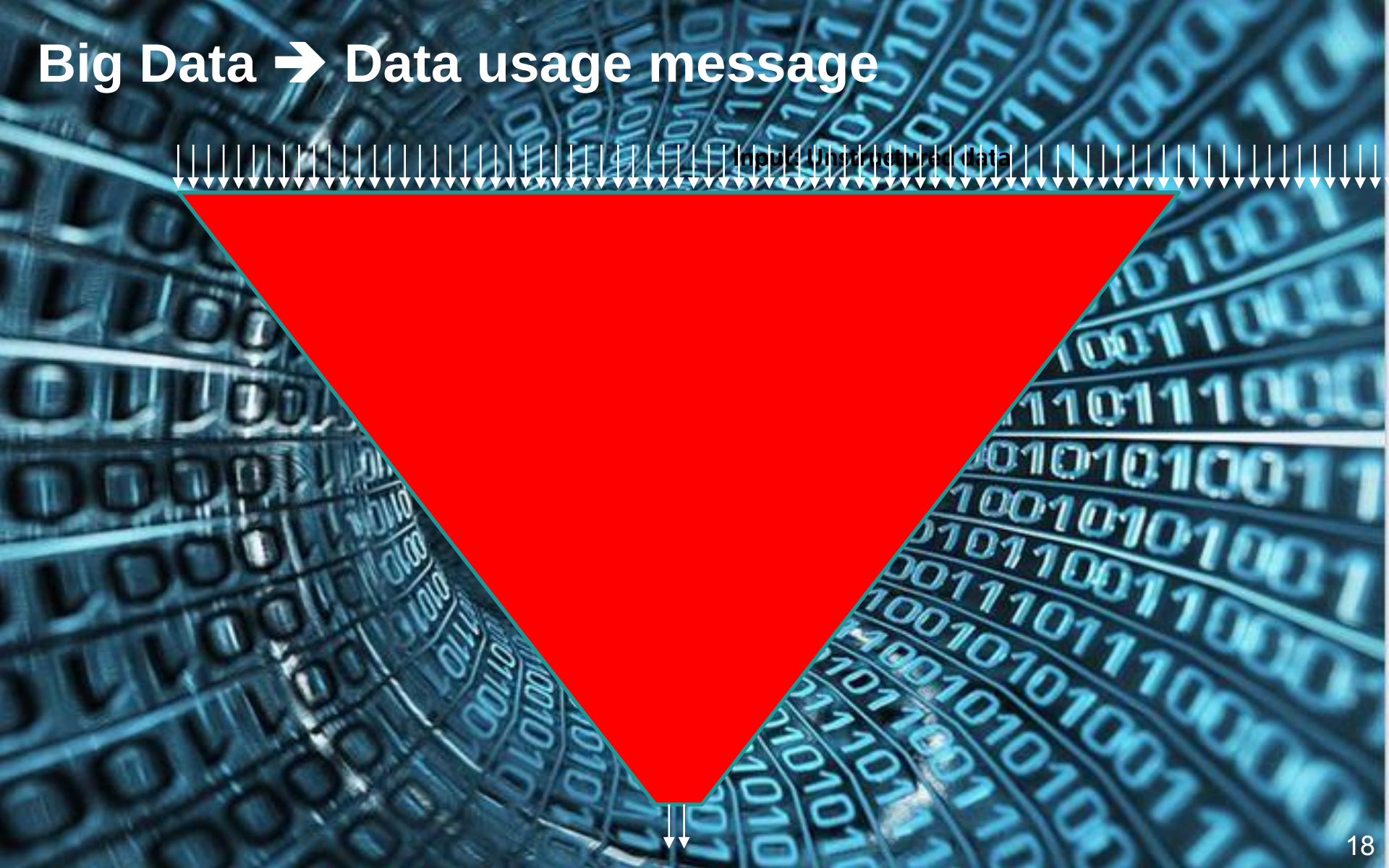


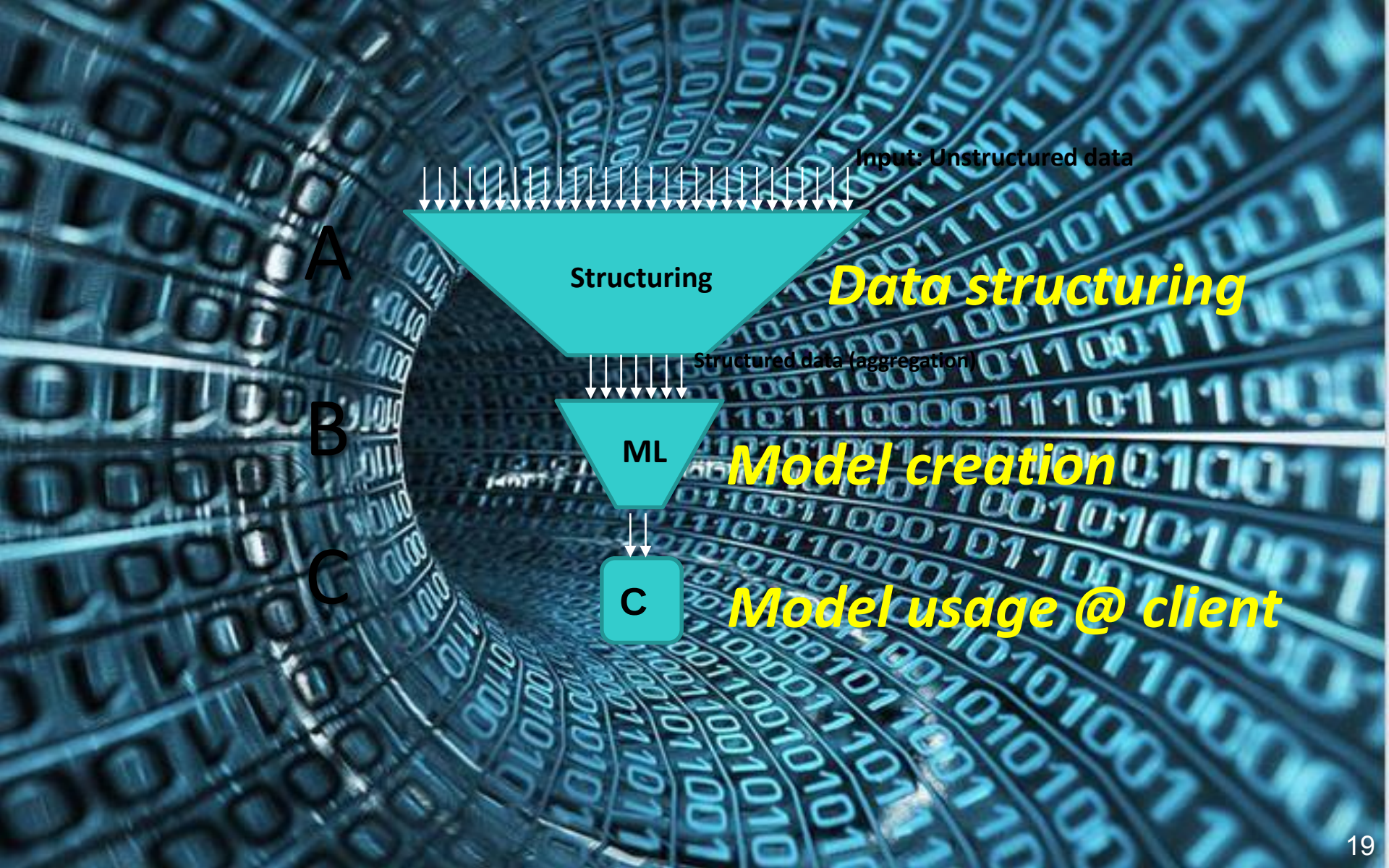
* At this point: unlimited BW to memory

Performance/Power 1/(Energy Per Instruction)

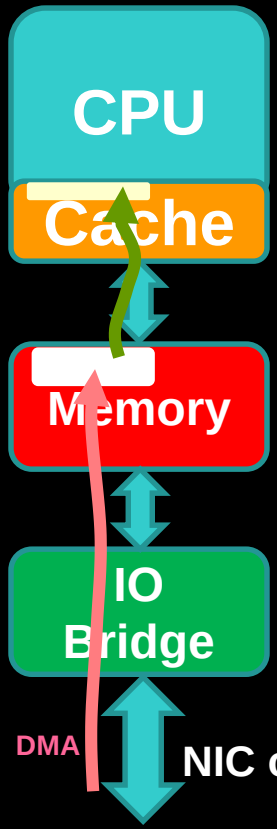


Big Data ➔ Data usage message




A
B
C
Input: Unstructured data
Structuring
Structured data (aggregation)
ML
C
Data structuring
Model creation
Model usage @ client

Existing Big data: Data movements



Copy of data

~nJoules/Byte

Cache/Memory are not effective if:

Cache related:
Reuse distance: >1M access

Memory related:
Reuse distance: >1G access

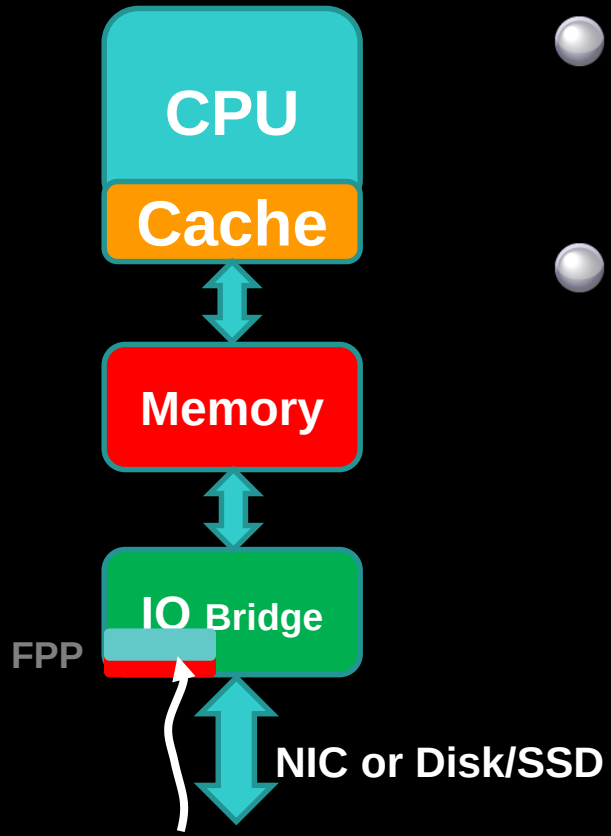
1. Why used-once data should move all the way to the “BIG” CPU?
2. Why use-once data is copied to memory?

Initial analysis: Hadoop-grep memory access

- Analysis of memory Hadoop-grep memory accesses was performed
- Unique addresses have been identifies
- In each pack (10M memory accesses), we counted;
  - number of unique addresses that have been single accessed
  - number of unique addresses that have been accessed multiple times
- About 50% of Hadoop-grep memory references have been single access

Big Data

Suggestion: Data movements reduction and free-up resources



- **Process Read-Once data close-to-IO**
  (Funnel PreProcessing FPP)
- **Implications:**
  - Free huge amount of memory for useful work (think Hadoop/Spark)
  - Process funnel functions by small efficient engines
  - Save Read/Write DRAM energy
  - Think about Big Data…

Open issues for research

- SW and OS
  - Co-Processor or
  - Heterogeneous system
- Compatibility
- Application awareness
- …

Summary

- The Funnel functions – execute close to the data source
  - Free up system's memory
  - Reduction of Data movement
  - Simple energy efficient engines at the front end
  - Issues
    - Compatibility issue: Apps, OS,
    - Amount of energy saving…
    - ….

Thank You