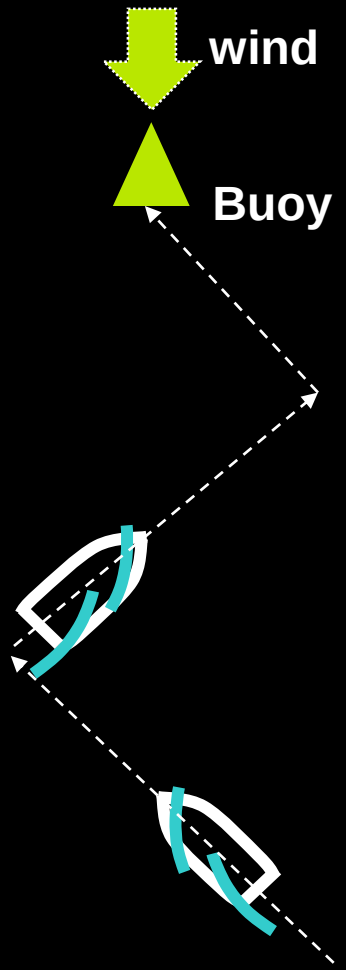
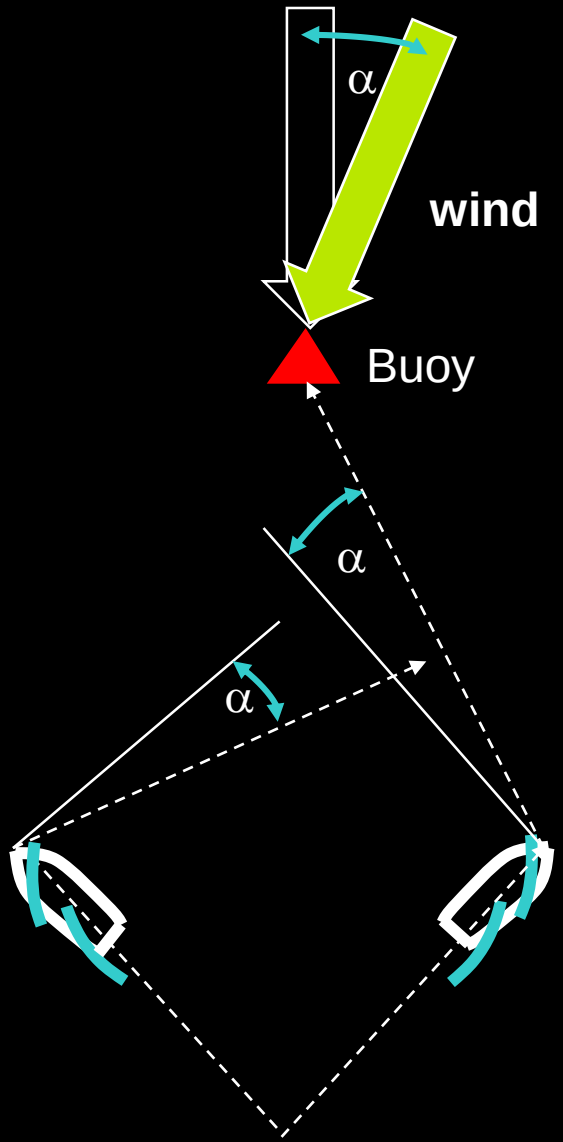
Sailing Basics



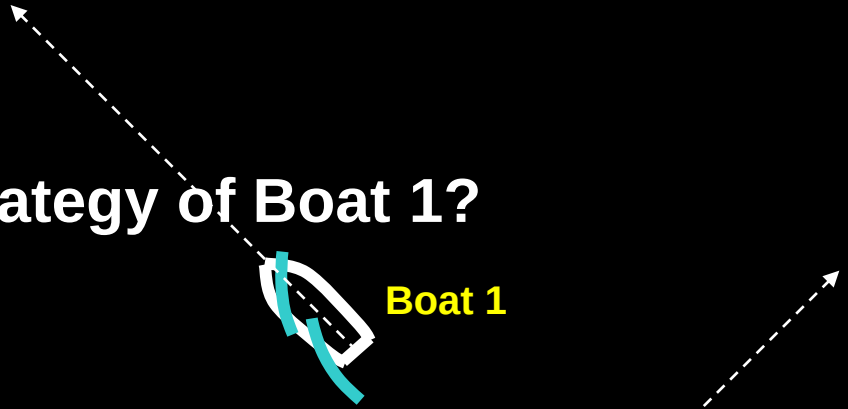
My moto:
Sailing - wind shift



Sailing competition

getting first

- What is the Strategy of Boat 1?



- What is the Strategy of Boat 2?

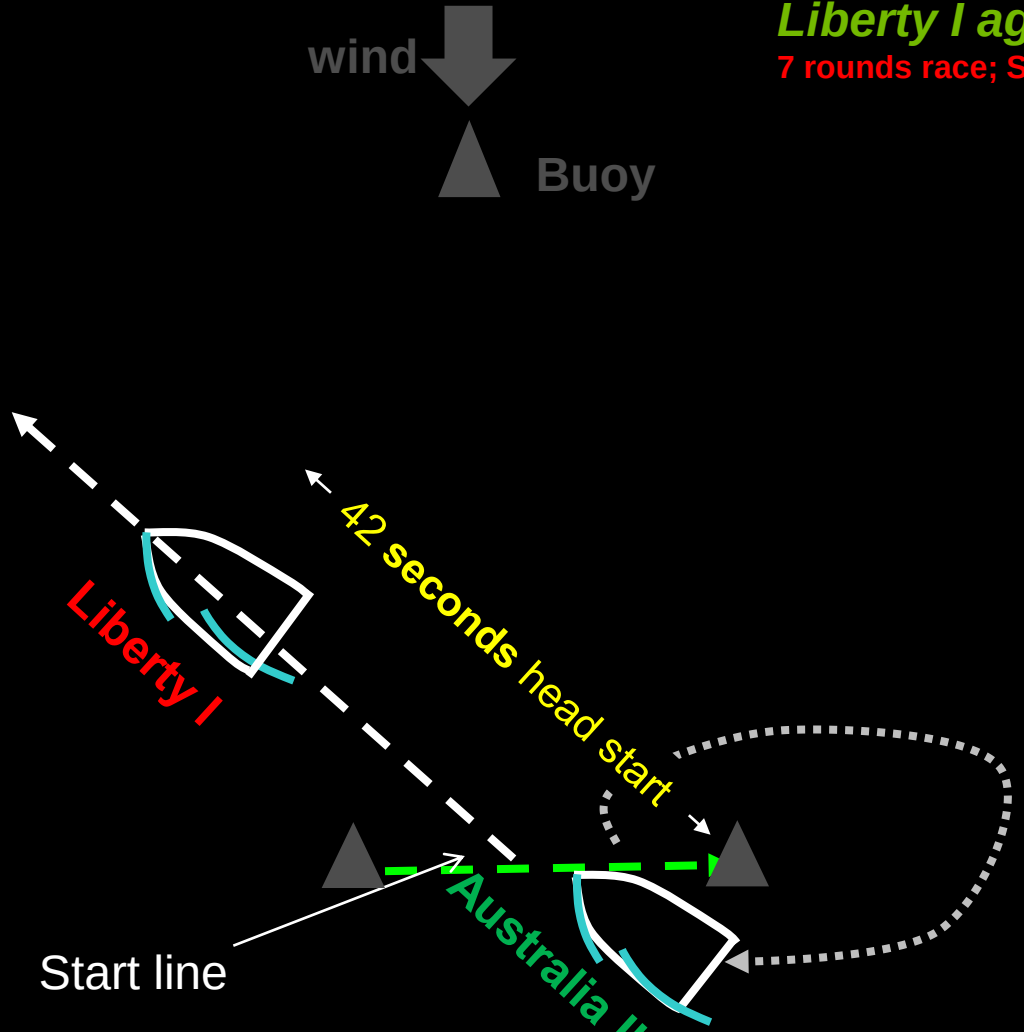
My Moto: Do not follow ➔ Invent

25th American Cap – September 23rd 1983

(Trophy was held by NYYC from 1857)

Liberty I against Australia II

7 rounds race; Status: Score 3:0 to Liberty I, 4th round started



Uri Weiser
Technion
Haifa, Israel

Potential future research in computing

Heterogeneous systems' optimization

memory subsystems - *Process-in-Storage when?*

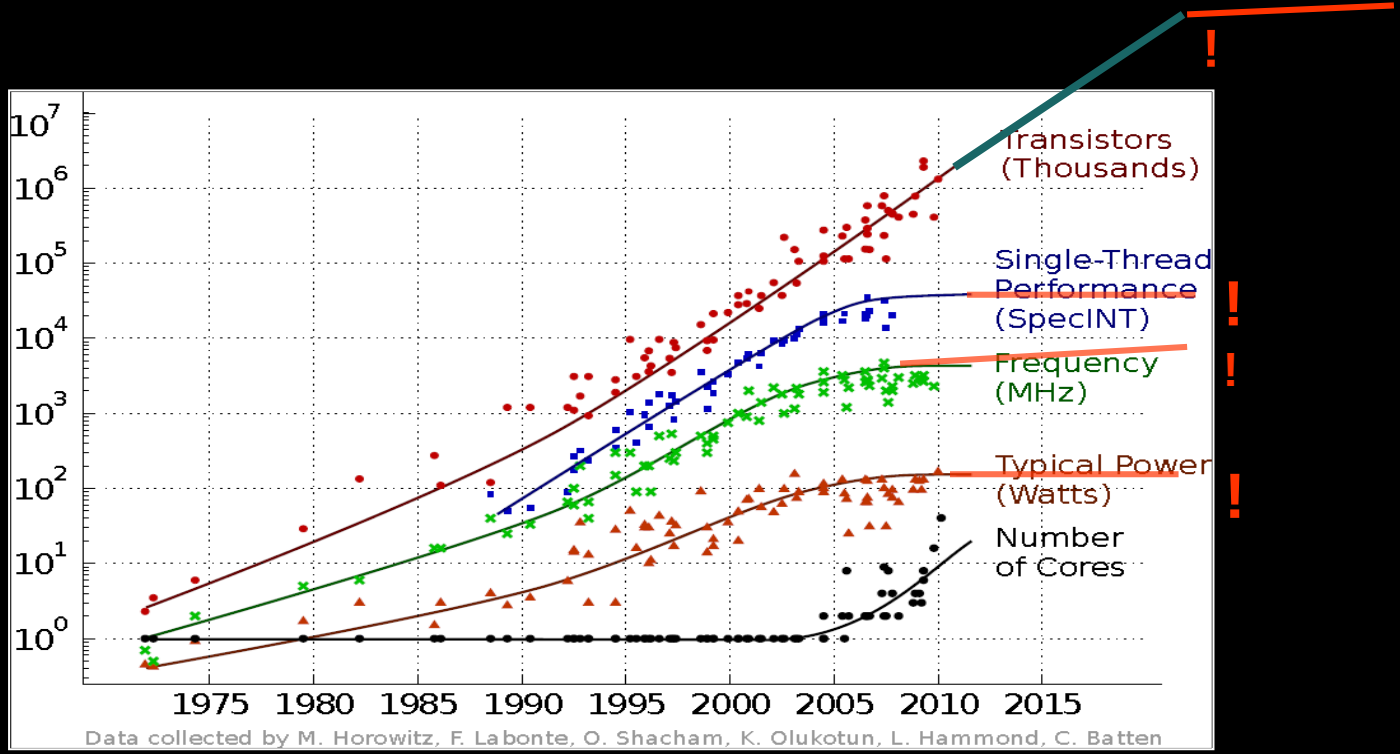
The talk covers research done by: Prof. Y. Etsion, Prof. I. Keidar, Prof. A. Kolodny, T. Morad, , Prof. A. Mendelson, G. Shmron, Prof. U. Weiser

The talk today

- The environment changes
  - Slow down in Moore's law
  - Slow down in the ability to enhance single core performance
    - ➔ Heterogonous systems
  - Big Data
    - Potential change in the way we handle data
    - ➔ new thinking about moving data?
- Heterogeneous system optimization
- Big Data --- data handling
  - Is it data movement?
  - Is it bandwidth?
- Example

Environment changes

- Slow down in process technology
- Slow down in Single thread core performance trend
- Power limitation



Environment changes

- Slow down in process technology
- Slow down in Single thread core performance trend
- Power limitation

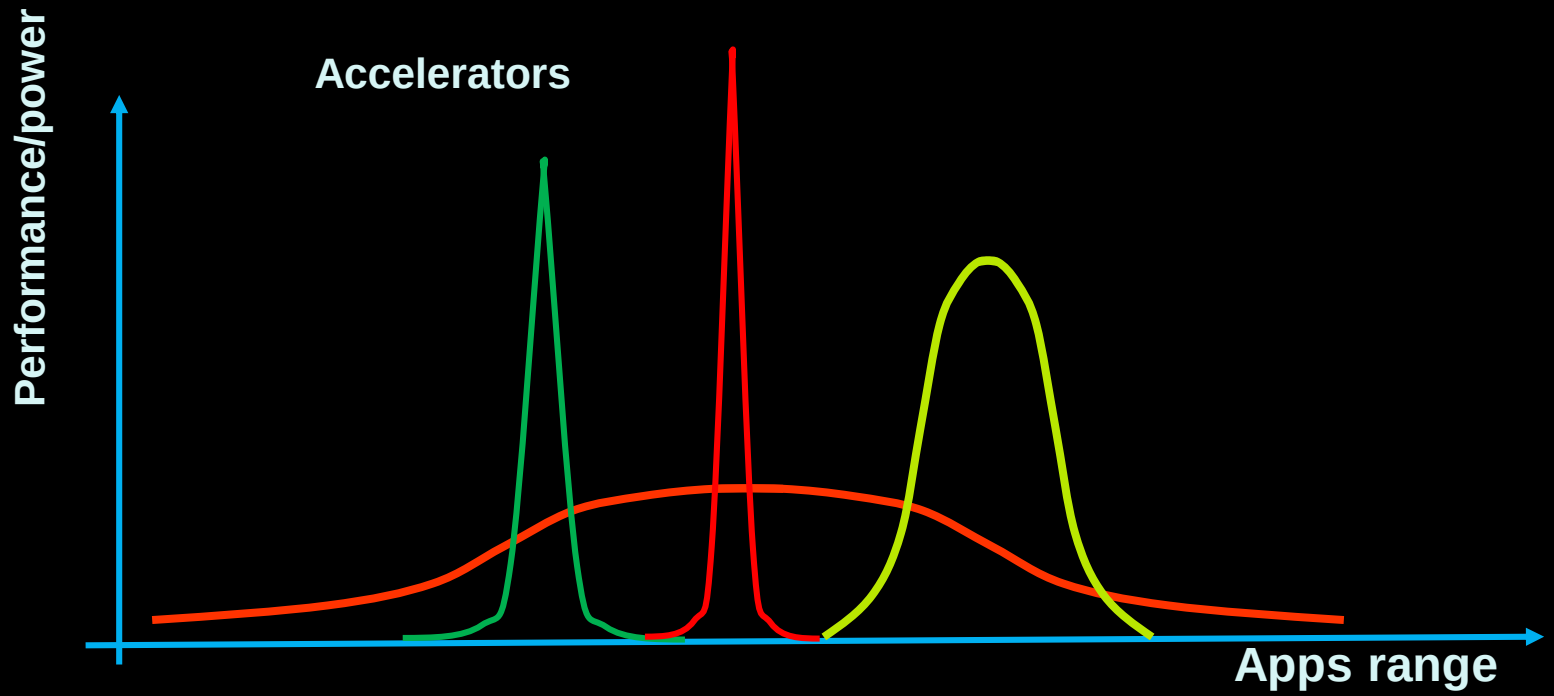
➔ **The Era of Heterogeneous systems** *(we are there already)*

- ***How to handle heterogeneousity?***
- ***Heterogeneous vs. general purpose engine?***
- **Size, power, energy, location of the accelerators?**
  - ➔ **Application phase specific**

A New Architecture Avenues?

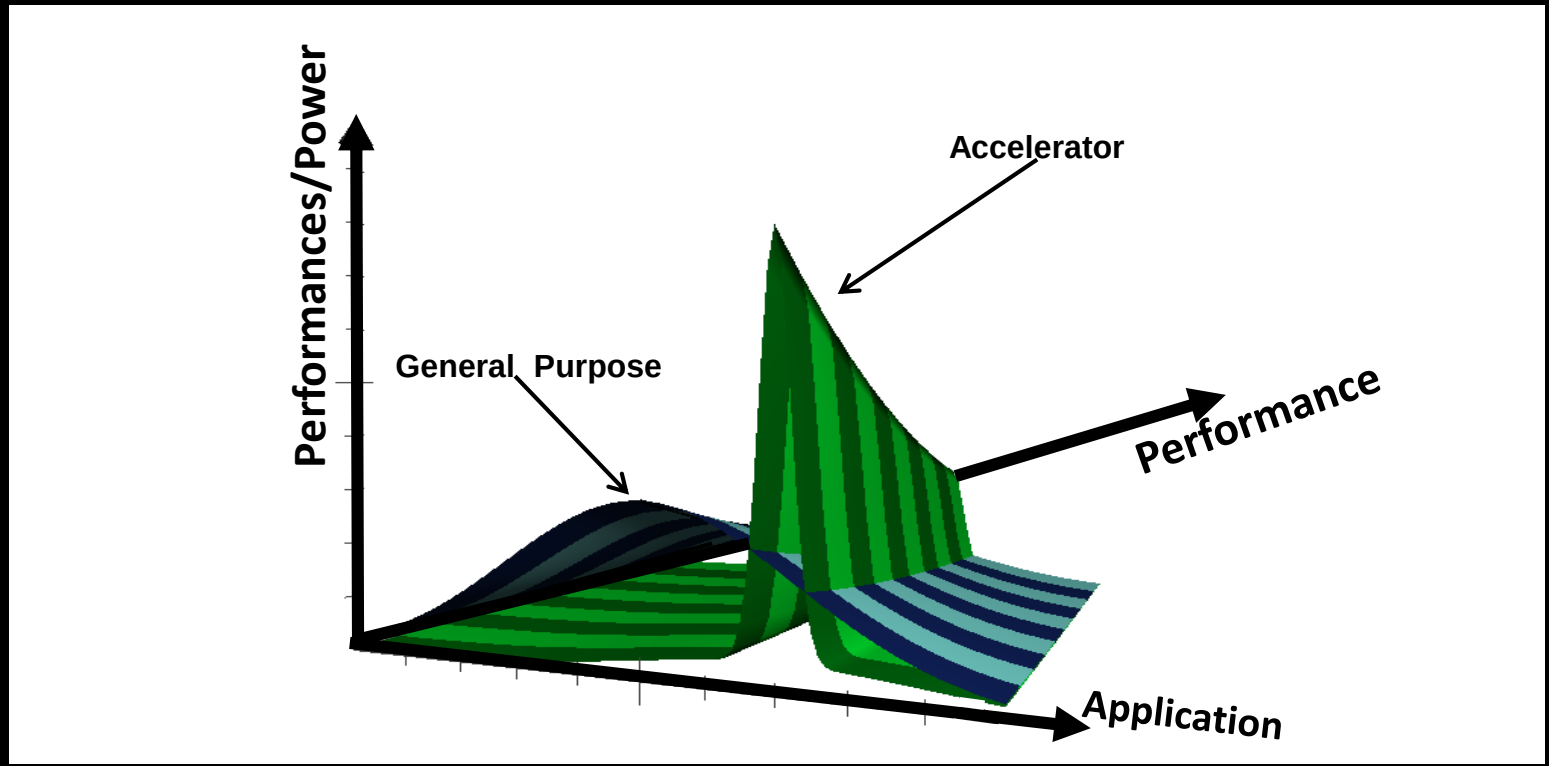
- Heterogeneous computing
  - The Era of Heterogeneous systems
  - HW/SW to fit application
  - Dynamic tuning
  - Accelerators
  - ➔ Optimizations: performance, energy efficiency
- Big Data = big
  - In general non repeated access to all the "Big Data"
  - What are the implications?

Heterogeneous computing



Continue performance trend by via Heterogeneous systems

Heterogeneous Computing



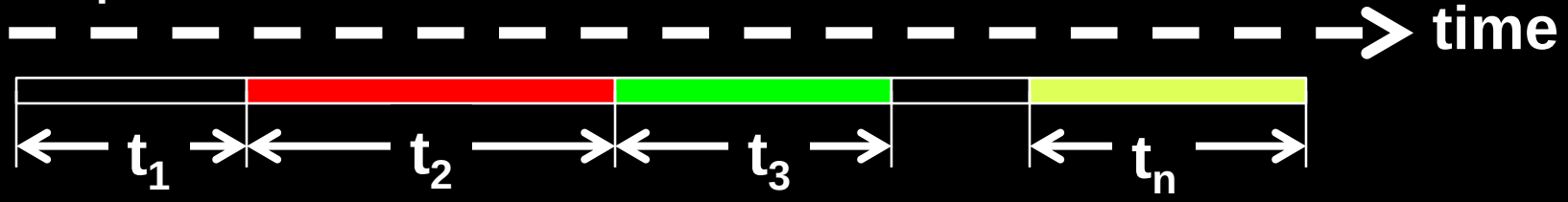
Heterogeneous Systems' Environment

- Environment with limited resources
- Need to optimize system's targets within resource constrains
- Resources may be:
  - Power, energy, area, space, $
- System's targets may be:
  - Performance, power, energy, area, space, $

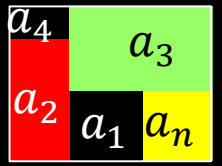
Heterogeneous system

- **Heterogeneous system design under resource constraint**
  how to divide resources (e.g. area, power, energy) to achieve maximum system's output (e.g. performance, throughput, energy savings)

Example:



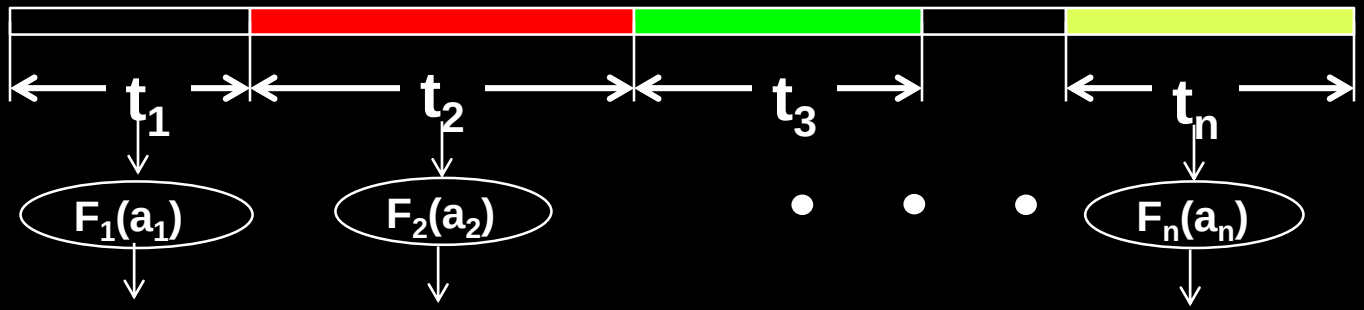
t_i = execution time of an application's section (run on a reference computing system)



$$A = \sum_{i=1}^{i=n} a_i$$

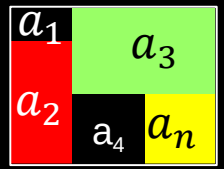
Accelerator target (an example): Minimize execution time under Area constraint

MultiAmdahl:



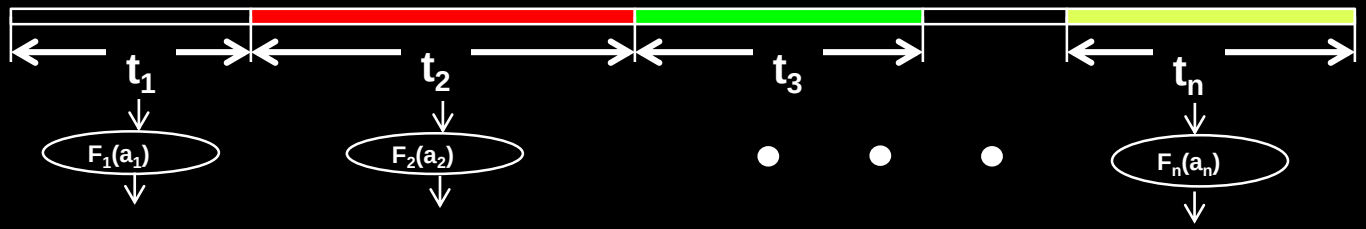
$T = t_{1*} F_1(a_1) + t_{2*} F_2(a_2) + \dots + t_{n*} F_n(a_n)$

$A = a_1 + a_2 + a_3 + \dots + a_n$



Target: Minimize T under a constraint A

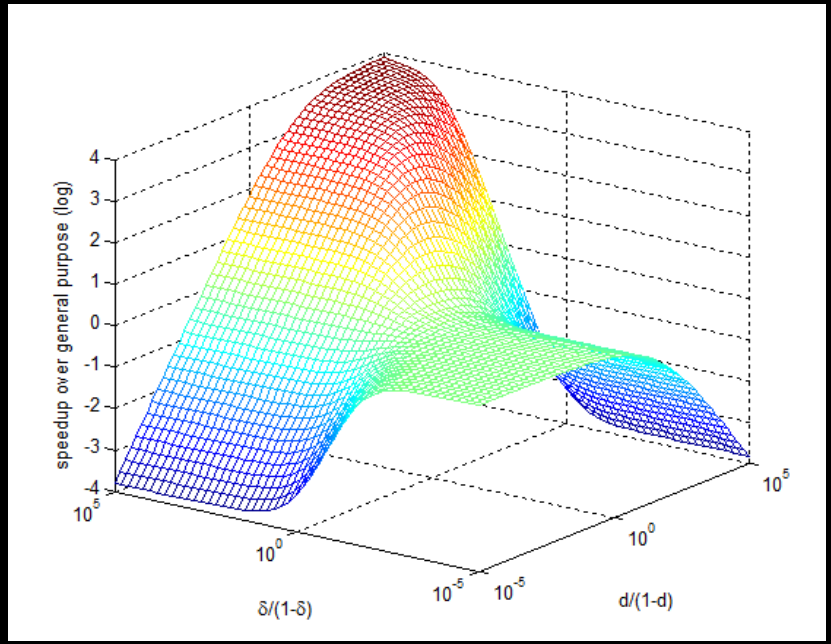
MultiAmdahl:



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

$$t_j F'_j(a_j) = t_i F'_i(a_i)$$

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



MultiAmdahl Framework

- Applying known techniques* to new environments
- Can be used during system's definition and/or dynamically to tune system

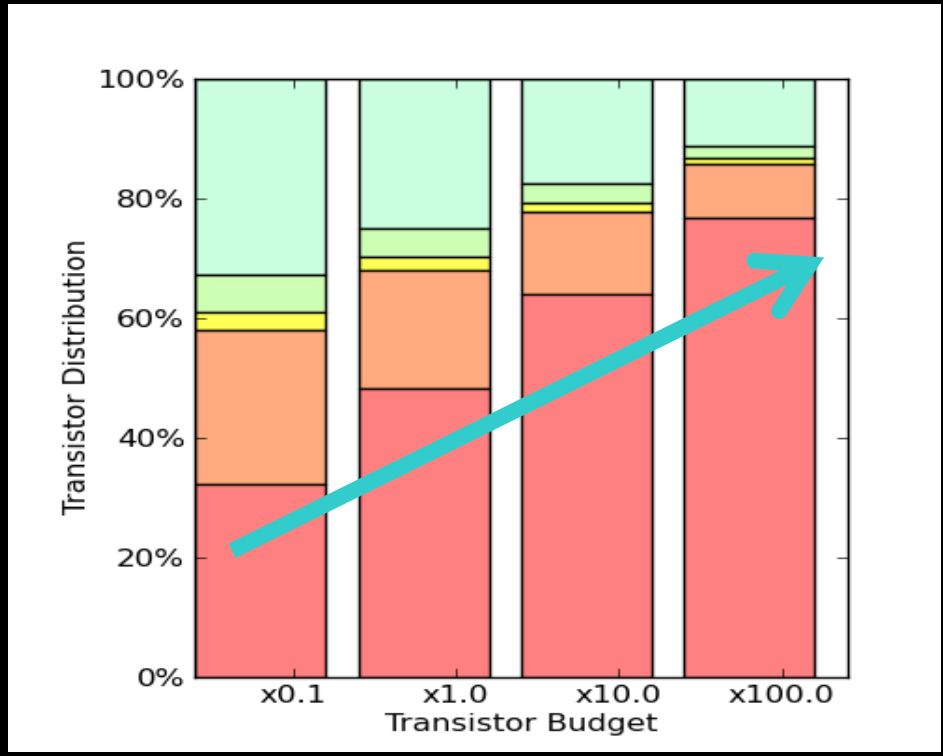
* Gossen's second law (1854), Marginal utility, Marginal rate of substitution (Finance)

Example: CPU vs. Accelerators

Future GP CPU size vs. transistor budget growth

Test case:
4 accelerators and GP (big) CPU

Applications: evenly distributed benchmarks mix w/ 10% sequential code



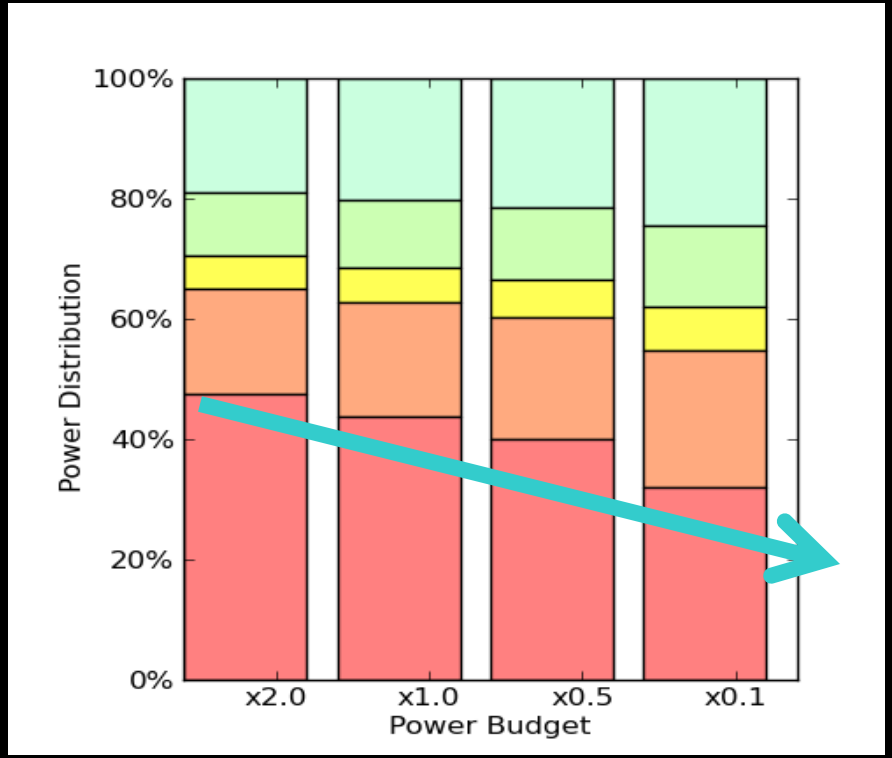
Heterogeneous Insight:
In an increased-transistor-budget-environment, General Purpose (big) CPU importance will grow

Example: CPU vs. Accelerators

GP CPU size vs. power budget

Test case:
4 accelerators and GP (big) CPU

Applications: evenly distributed benchmarks mix w/ 10% sequential code



Heterogeneous Insight:
In a decreased-power-budget-environment, Accelerators importance will grow

What is $F(a_i)$?

- You can look at it as the acceleration vs. area (or energy, power etc.) BUT
- There are more parameters that impact the function $F(a_i)$ e.g. LOCATION*

* example as part of the “Process In Storage”

Big Data

- 90% of data in the world has been created in the last 2 years
- By 2020 world's data will grow 50 times from today
- ➔ Change the way we handle
- ➔ Big Data processing

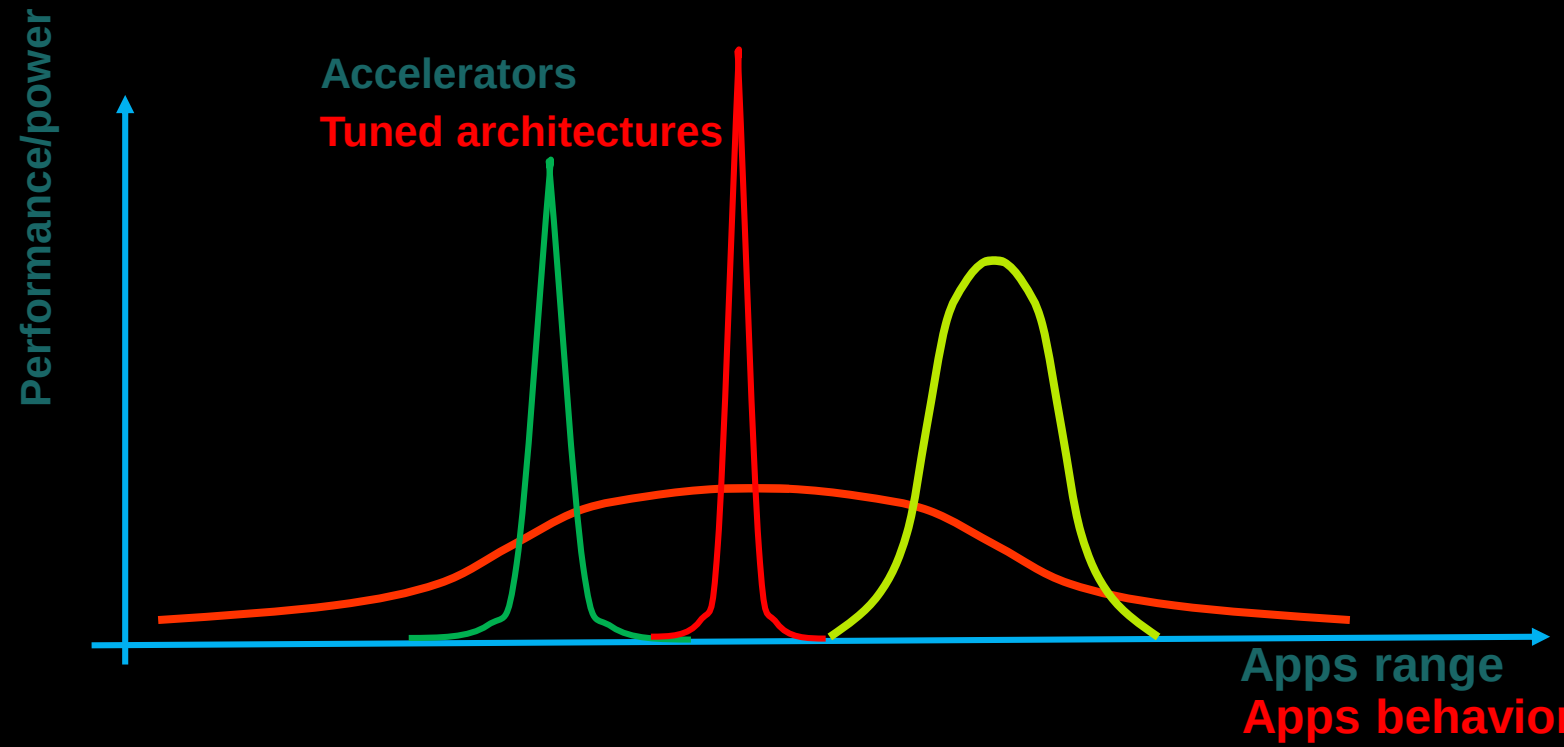
Big Data Environment

- Big Data = big
  - In most of the cases some of the data is irrelevant (Extract Transform and Load (ETL)) for the solution or its relevancy is simple (e.g. wordcount)
  - In general there is a non repeated access to all the “Big Data”
  - What are the implications?

A New Architecture Avenues in Big Data Environment

- Heterogeneous computing – ”tuning” HW to respond to specific needs
- example: Big Data memory access pattern
- Potential savings
  - Data Movements
  - Bandwidth

Heterogeneous computing : Application Specific Accelerators



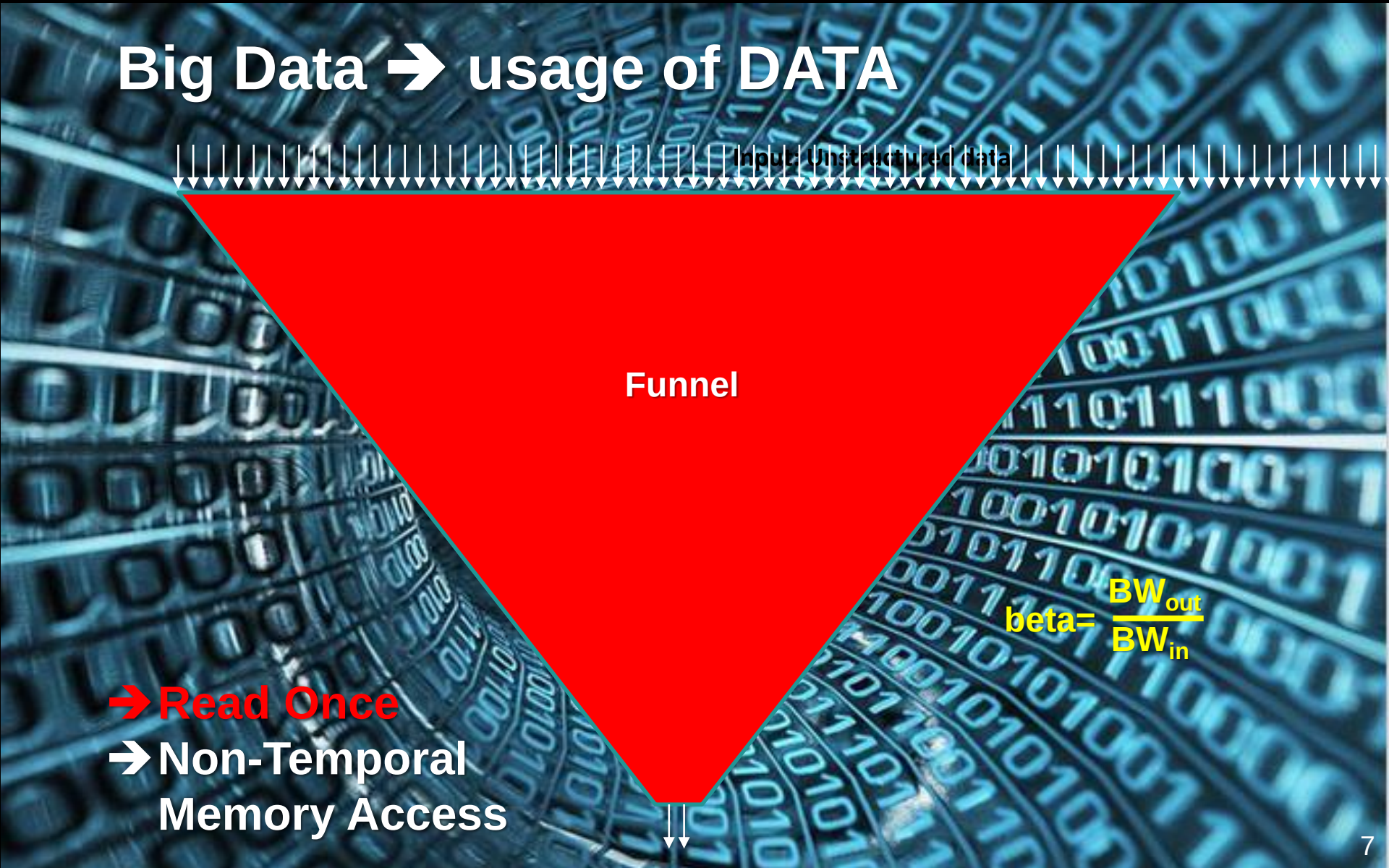
Continue performance trend by tuned architecture to bypass current technological hurdles

Reduction of system's energy in Big Data environment

- ❑ understand where energy is wasted
- ❑ Identify the energy hungry parts and performance bottleneck
- ❑ Provide a TAILORED solution for Data Center* usage

* It would not be simple

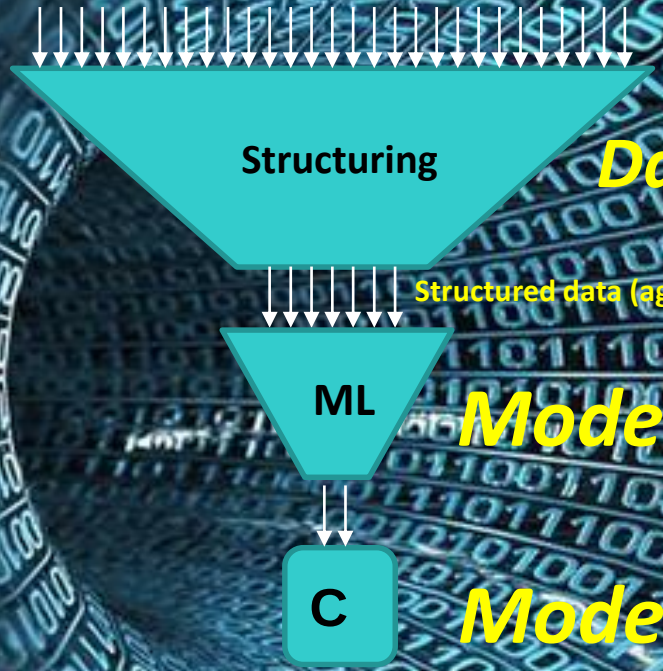
Big Data ➔ usage of DATA



➔ Read Once

➔ Non-Temporal Memory Access

Machine Learning



A

B

C

Data structuring = ETL

Model creation

Model usage @ client

Does Big Data exhibit special memory access pattern?

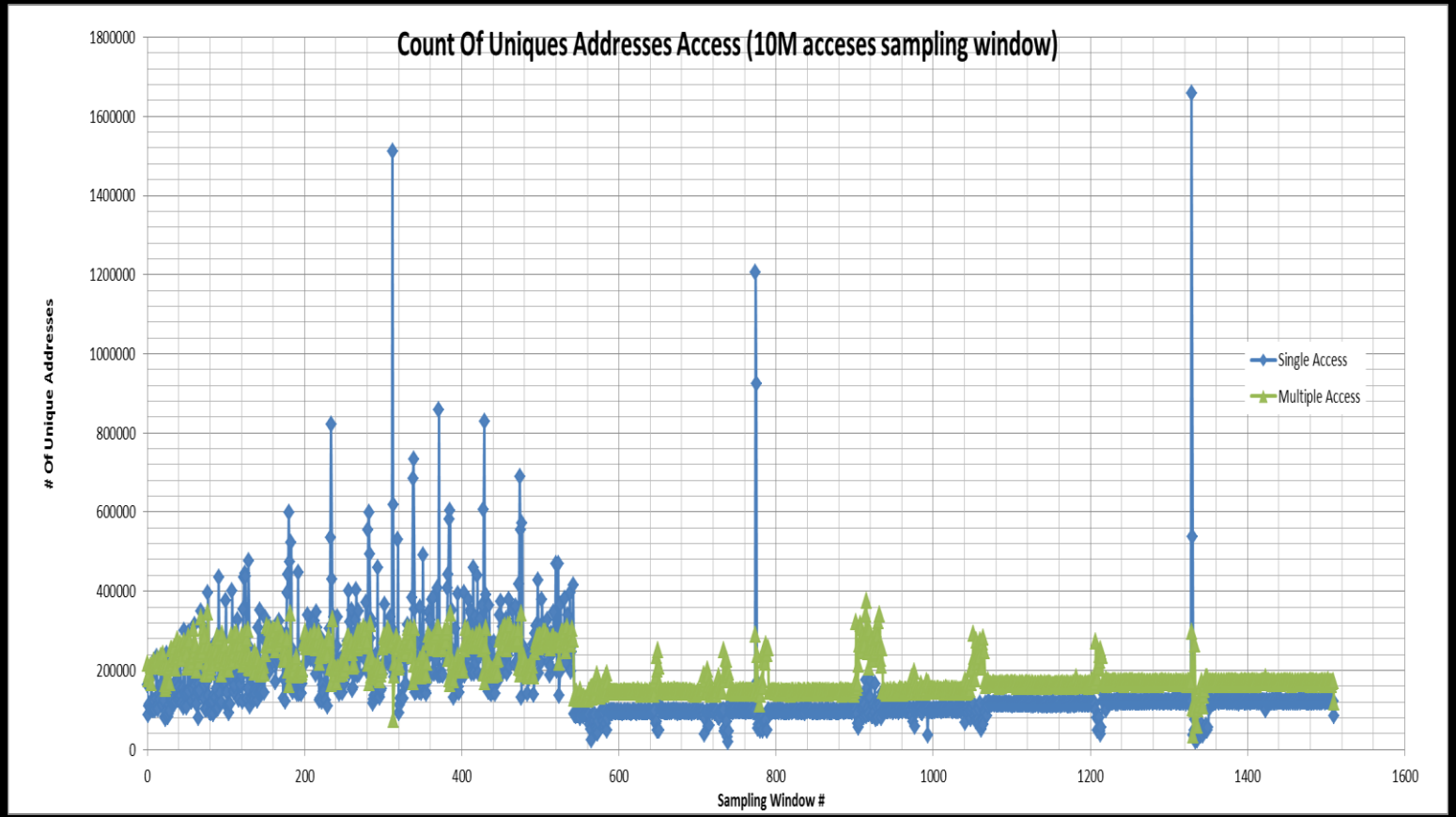
It probably should since

- Revisiting ALL Big Data items will cause huge/slow data transfers from Data sources
- There are 2 access modes of memory operations:
  - Temporal Memory Access
  - Non-Temporal Memory access
- Many Big Data computations exhibit a Non-Temporal Memory-Accesses and/or Funnel operation

Non-Temporal Memory access

Initial analysis: Hadoop-grep Single Memory Access Pattern

~50% of Hadoop-grep unique memory references are single access

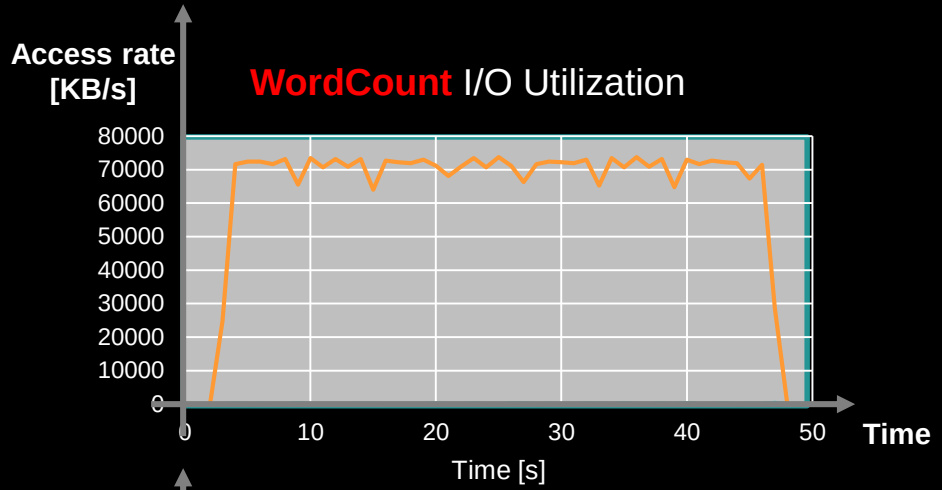


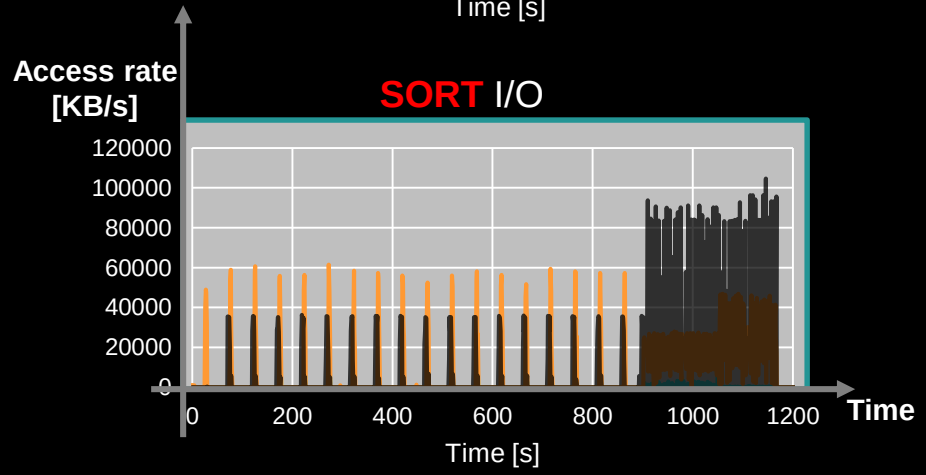
Non-Temporal Memory Accesses

Preliminary Results

- **WordCount:**
  Access to Storage:
  Non-temporal locality

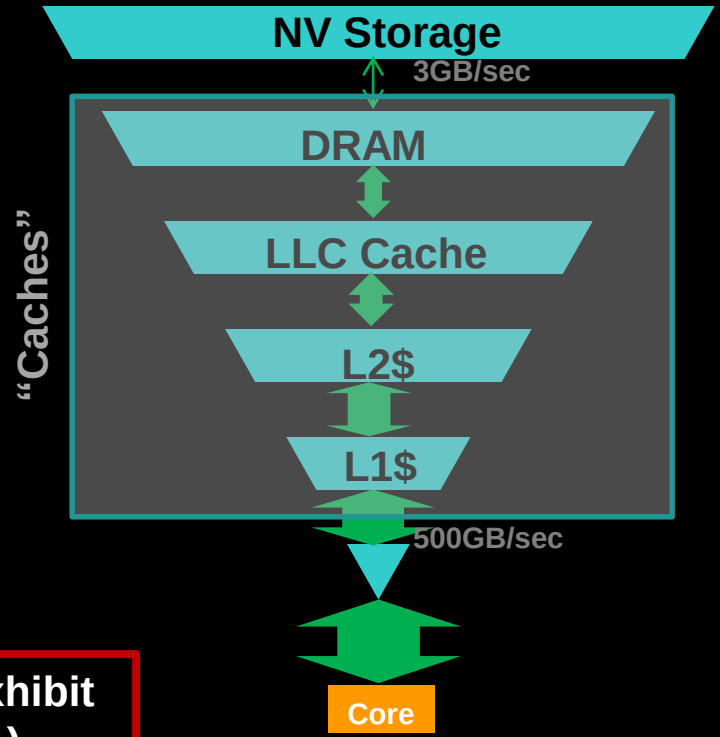
- **Sort:**
  Access to Storage:
  NO Non-temporal locality





Current systems

- Memory subsystem is tuned for “Temporal Memory Access”
  - ❑ DRAM – tuned for repeated page access
  - ❑ Cache – tuned for repeated cache block access



However, many Big Data applications exhibit Non-Temporal Memory Accesses (NTMA)

Where energy is wasted?

- **DRAM**
- **Limited BW**

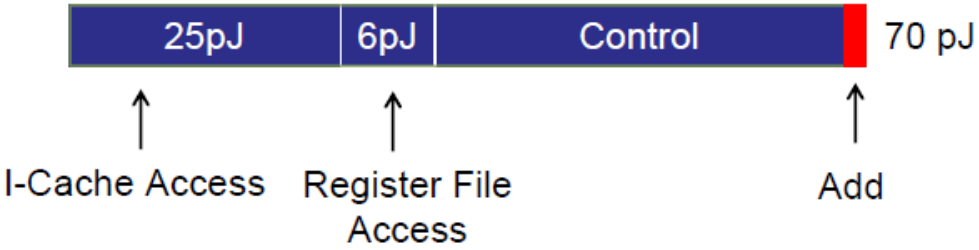
Rough Energy Numbers (45nm)

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3 pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

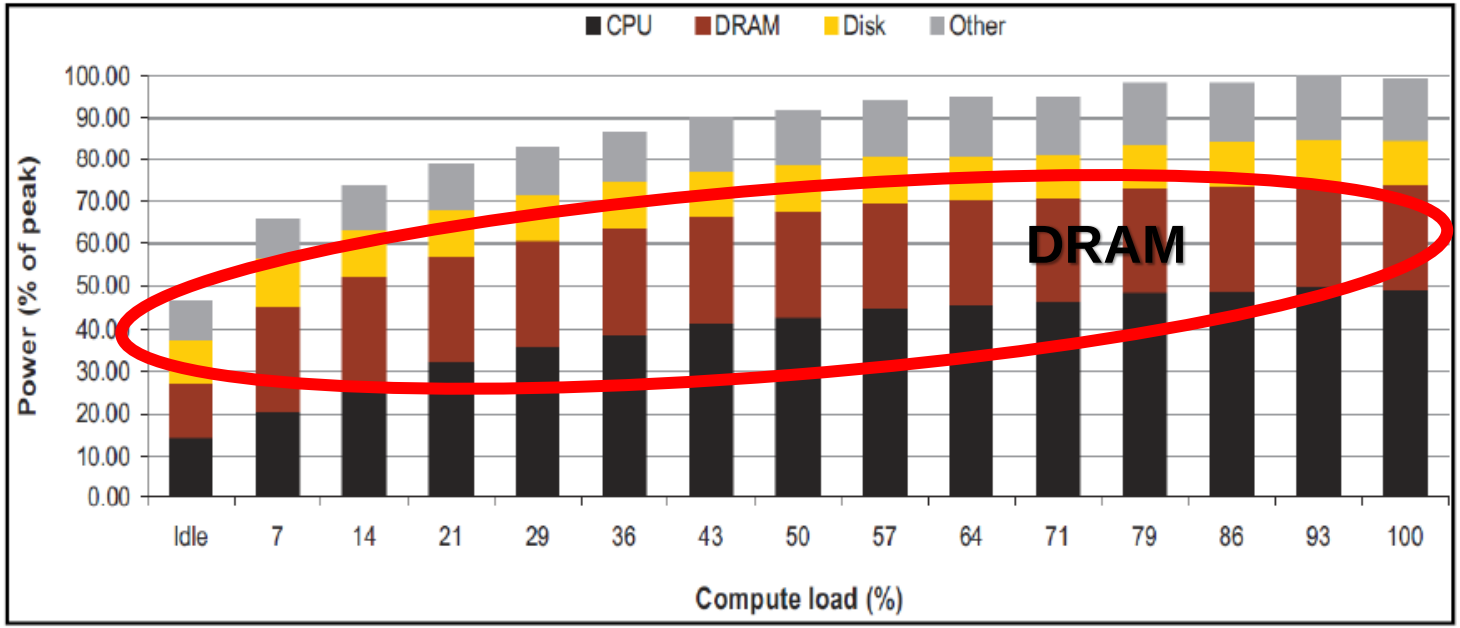
Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

Instruction Energy Breakdown



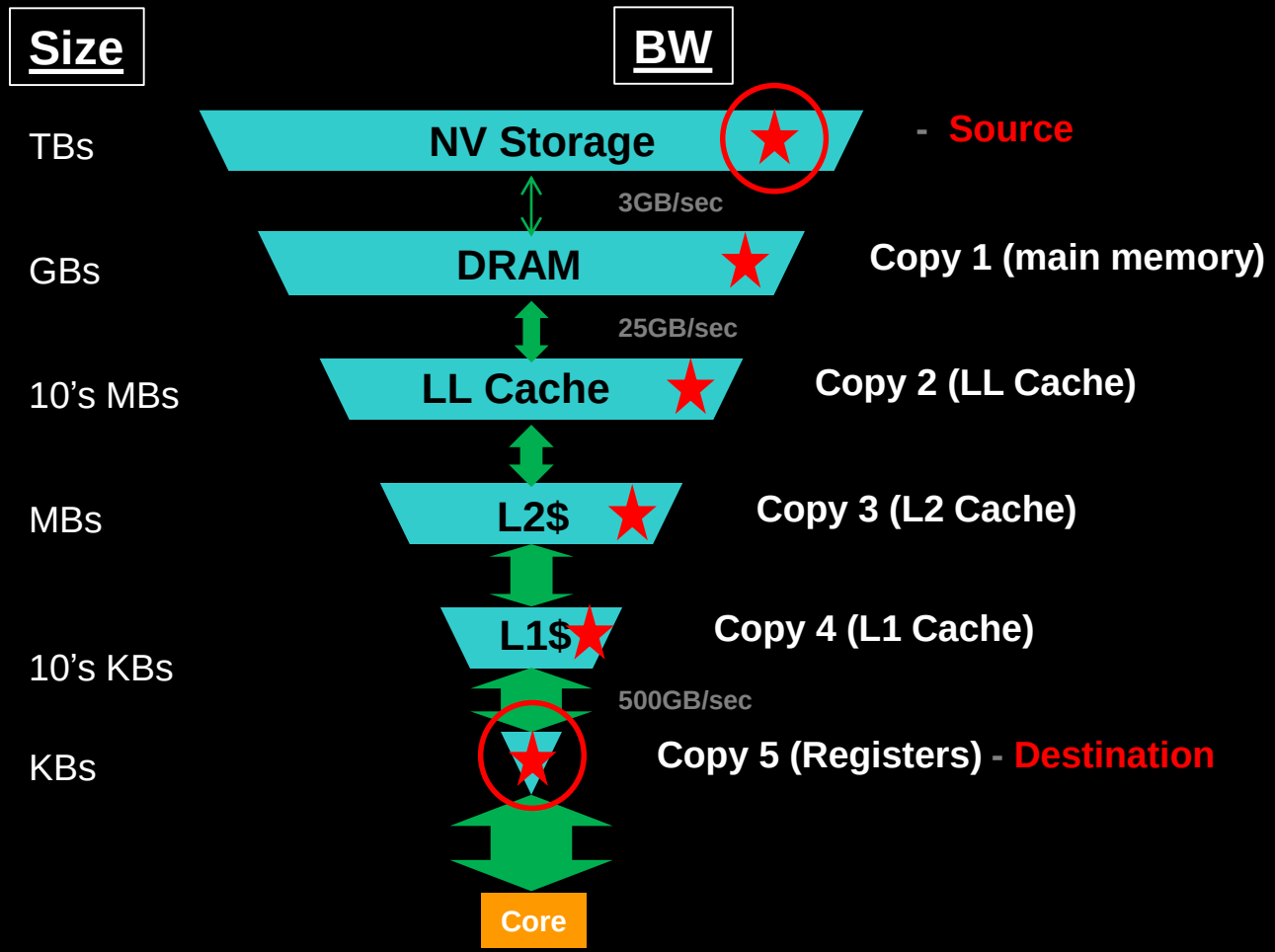
From: Bill Dally (nVidia and Stanford), Efficiency and Parallelism, the challenges of future computing

Data Center Energy Specs

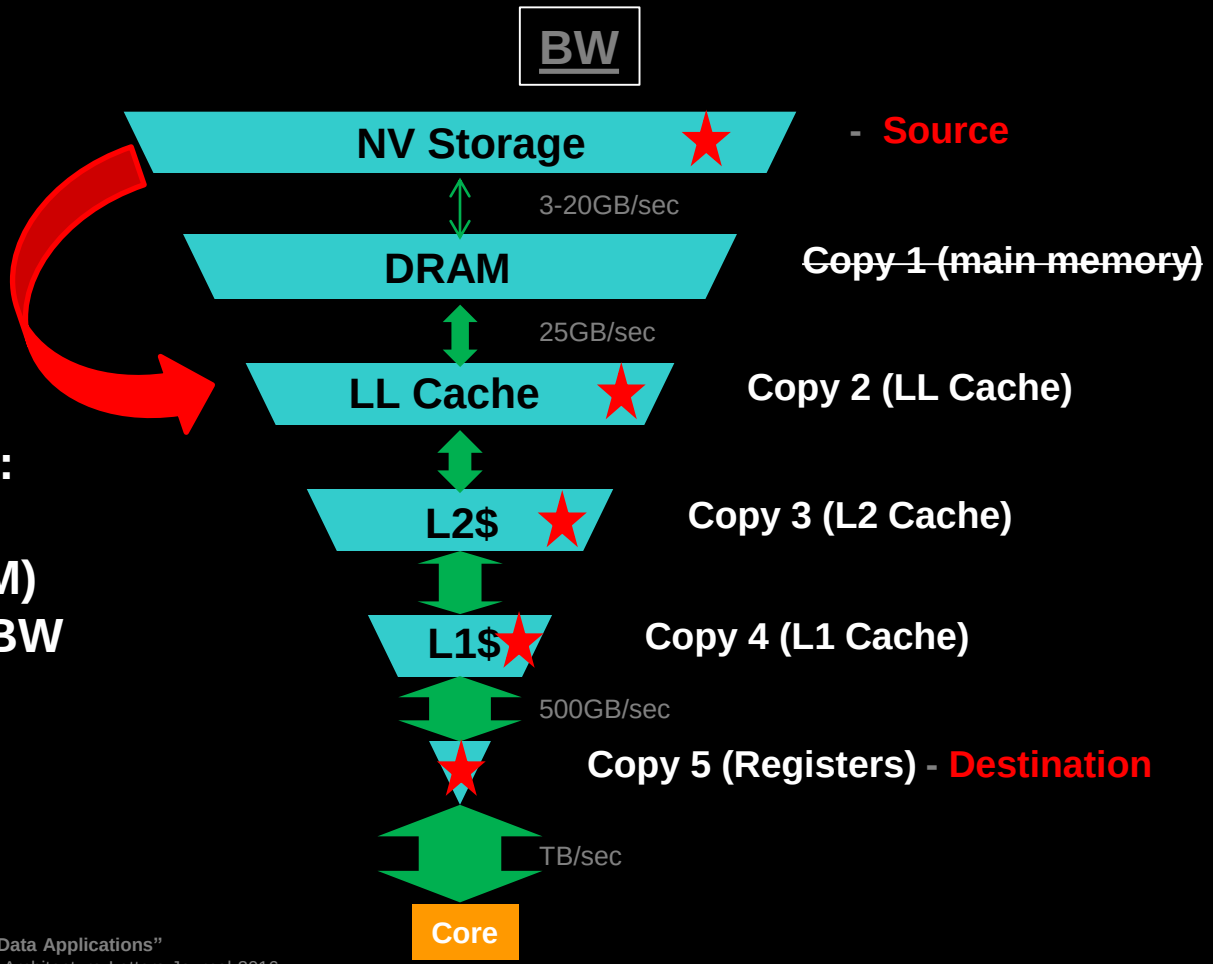


Malladi, ISCA, 2012

Memory Subsystem - copies



Memory Subsystem – DRAM bypass == DDIO



Potential savings:

@ 0.5n J/B (DRAM)
10 – 20 GB/s NV BW

➔ **5W – 10W**

Reference: "Optimizing Read-Once Data Flow in Big-Data Applications"
Morad, Ghomron, Erez, Weiser, Kolodny, in Computer Architecture Letters Journal 2016

Initial Experiment

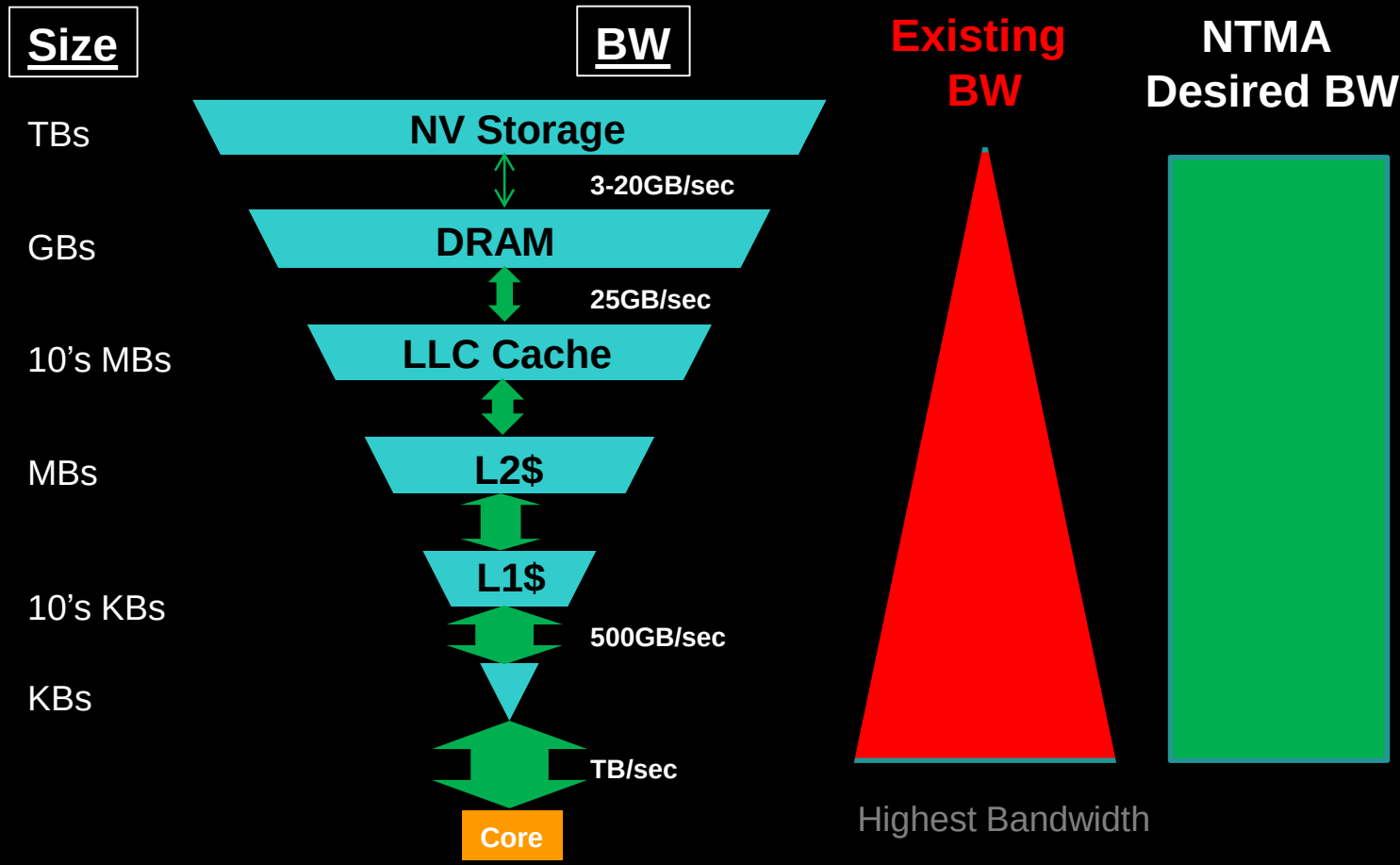
- Example program: read file from disk and XOR all values
- DDIO-aware code on a real system
  - Small buffer (fit into 2 ways of LLC)
  - Low latency from write to read (avoid evictions)
  - Zero-copy (O_DIRECT flag)
  - Bypass OS page cache (O_DIRECT flag)
  - Run code on chip that is connected to the SSD (OS affinity)
- Compare system with DDIO enabled and DDIO disabled
- Measure runtime, power and energy

Bandwidth

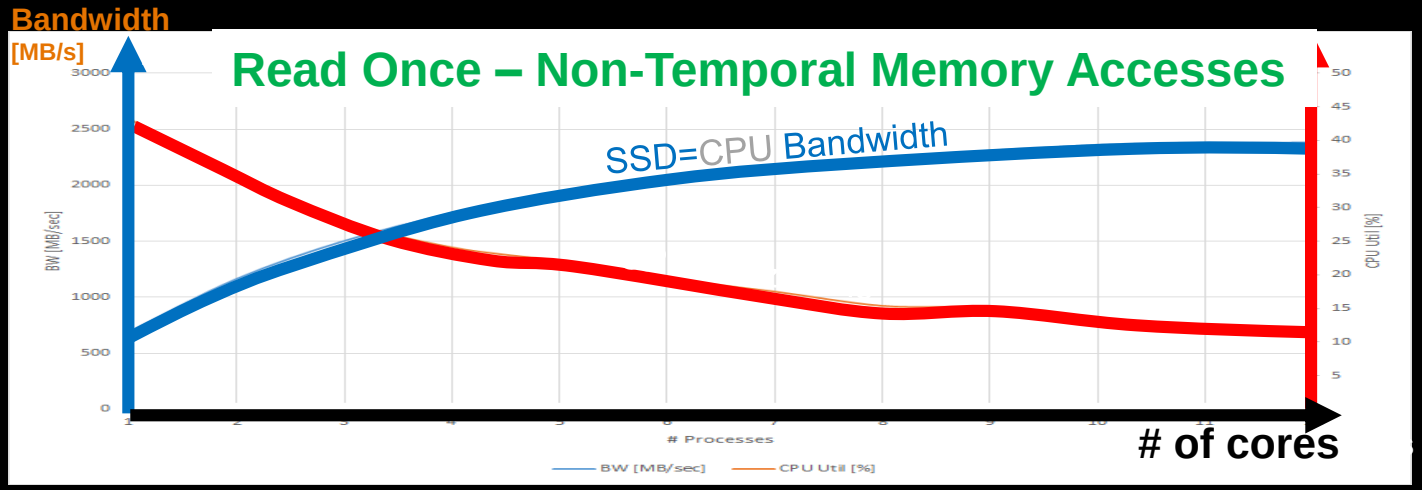
When should we use Funnel at the Data source

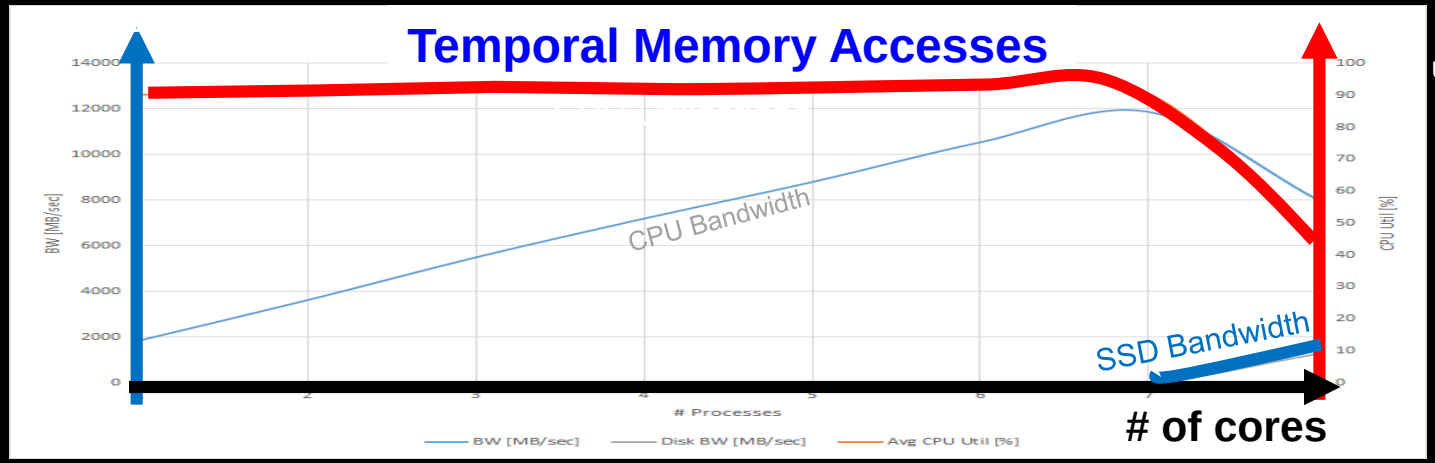
Memory Hierarchy is Optimized for

A: Bandwidth issue ➔ System are built for Temporal Locality



B: Memory access per operation impact BW



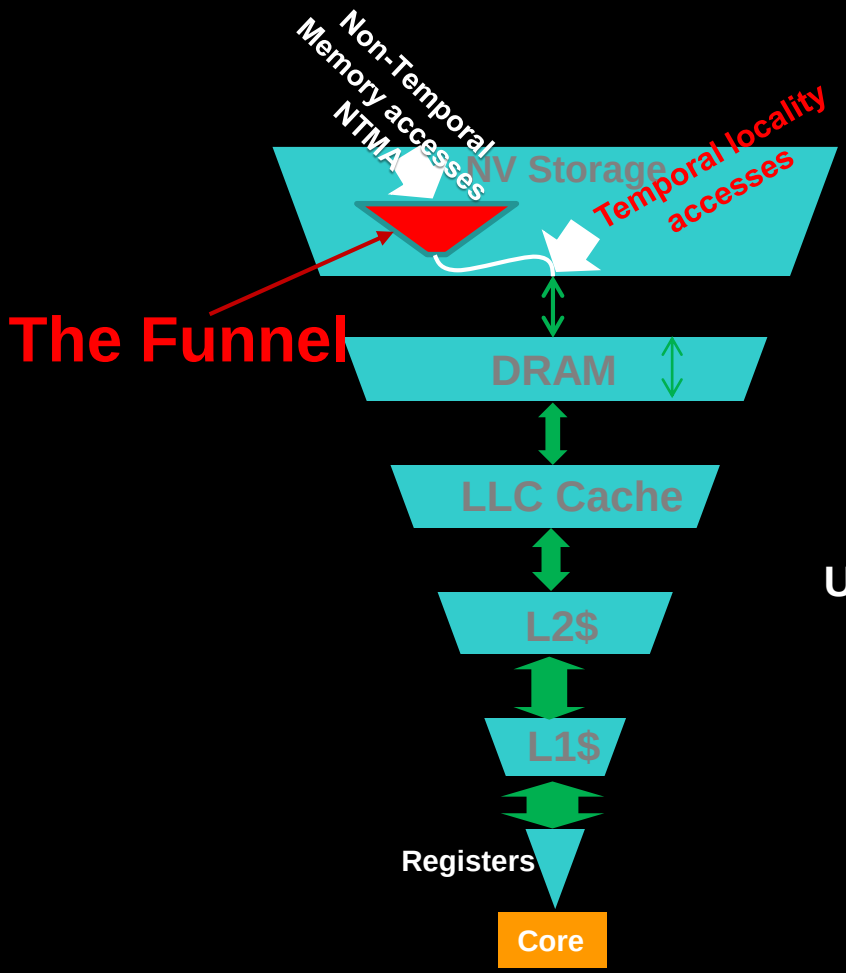


INITIAL RESULTS

Hint: Memory access per operation

Solution:

Flow of “Non-Temporal Data Accesses”

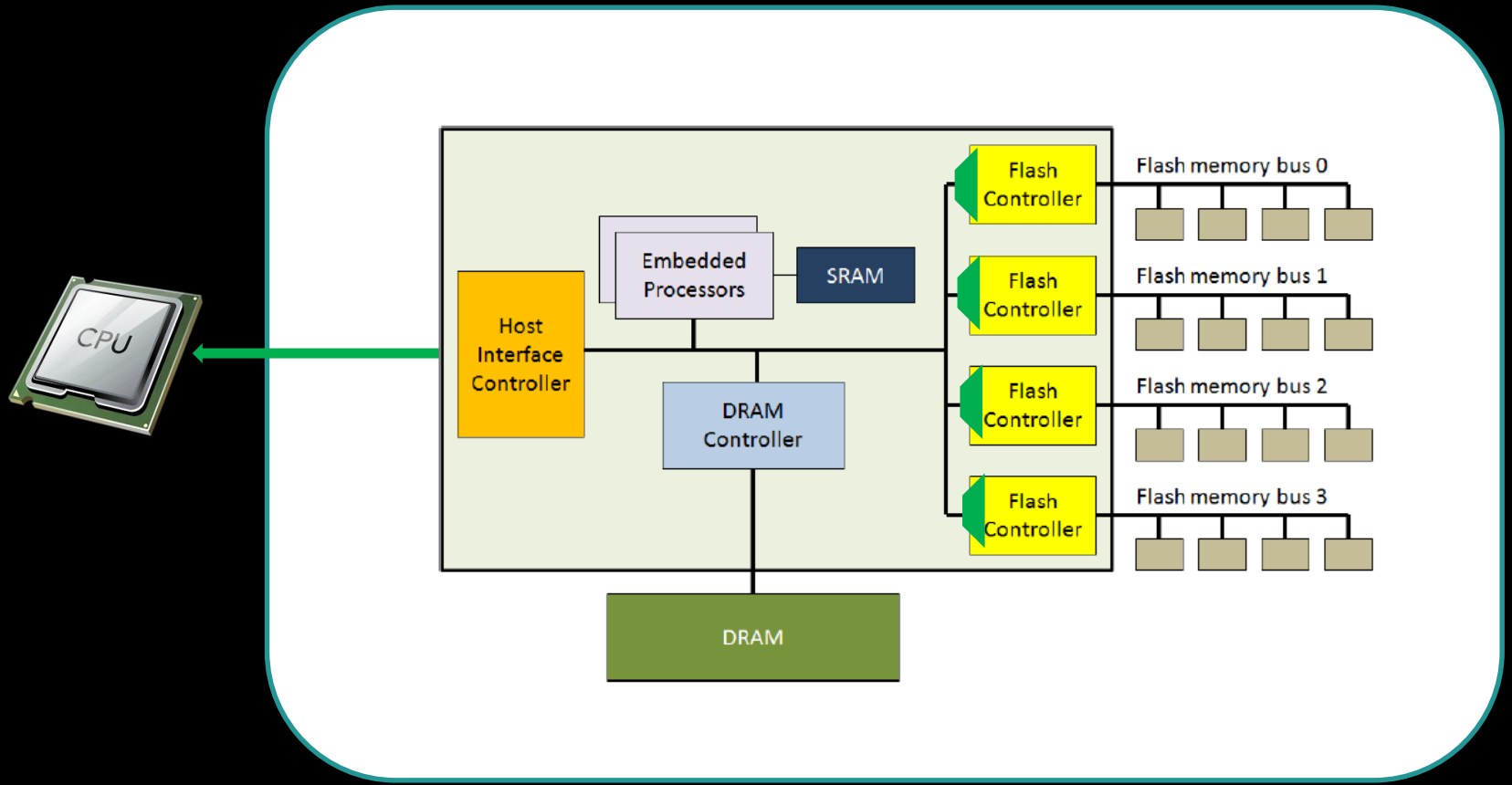


Use Funnel when Bandwidth bottleneck occurs➔

- “high” memory accesses per Instruction
- Limited BW
- Non temporal locality memory access

*private communication with: Moinuddin Qureshi

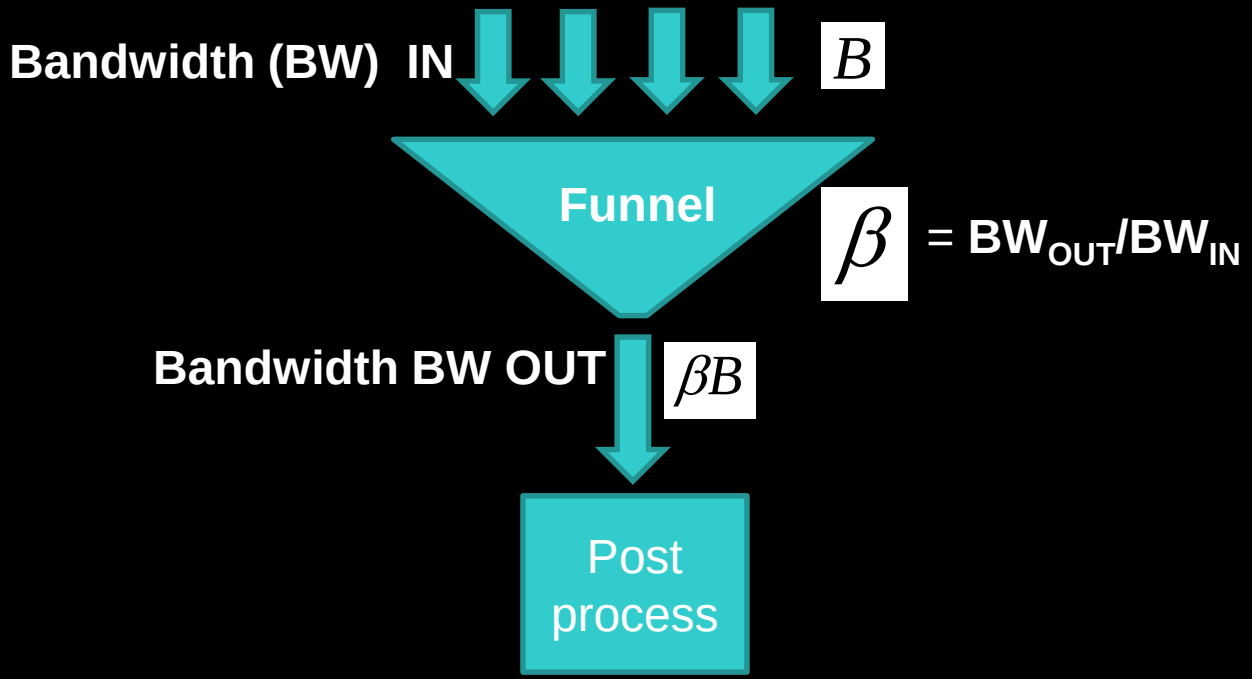
“Funnel”ing “Read-Once” data in storage



*Kang, Yangwook, Yang-suk Kee, Ethan L. Miller, and Chanik Park. "Enabling cost-effective data processing with smart ssd." In Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, pp. 1-12. IEEE, 2013.

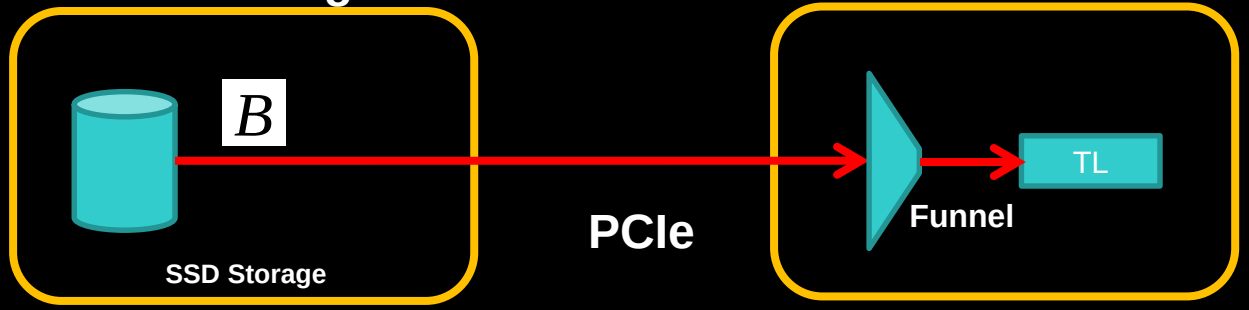
**K. Eshghi and R. Micheloni. “SSD Architecture and PCI Express Interface”

Analytical model of the Funnel



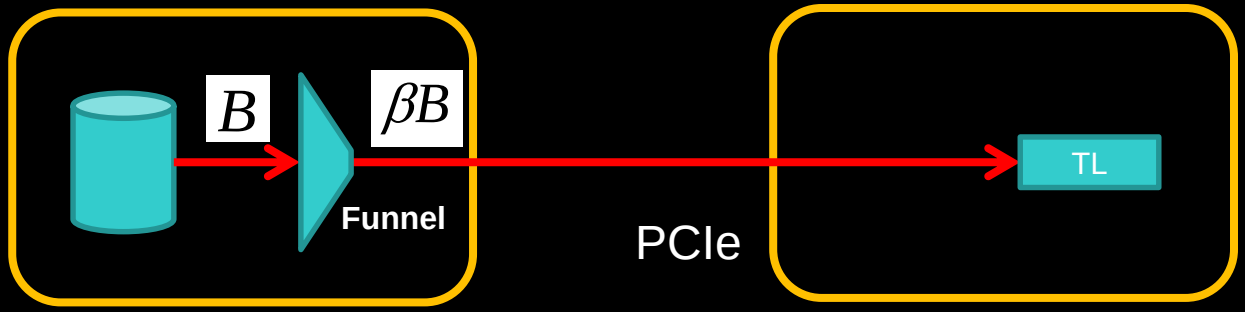
Purposed Architecture

Baseline Configuration

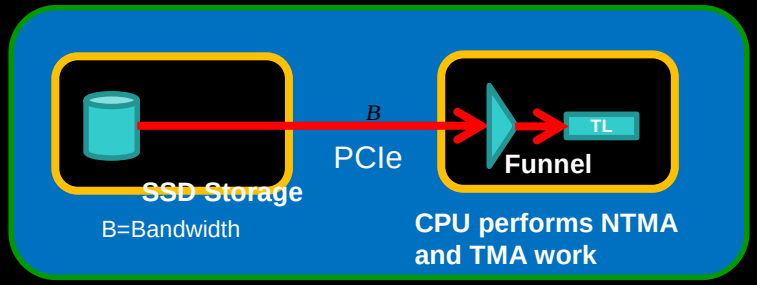


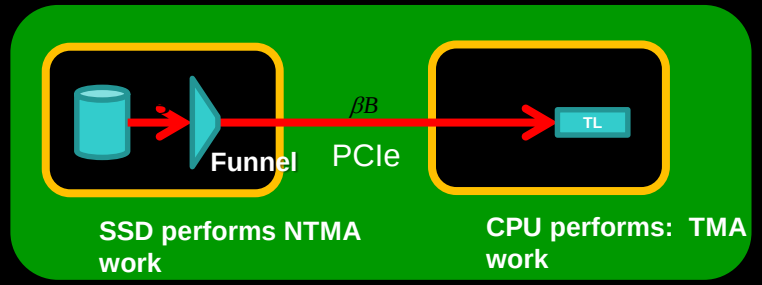
B=Bandwidth

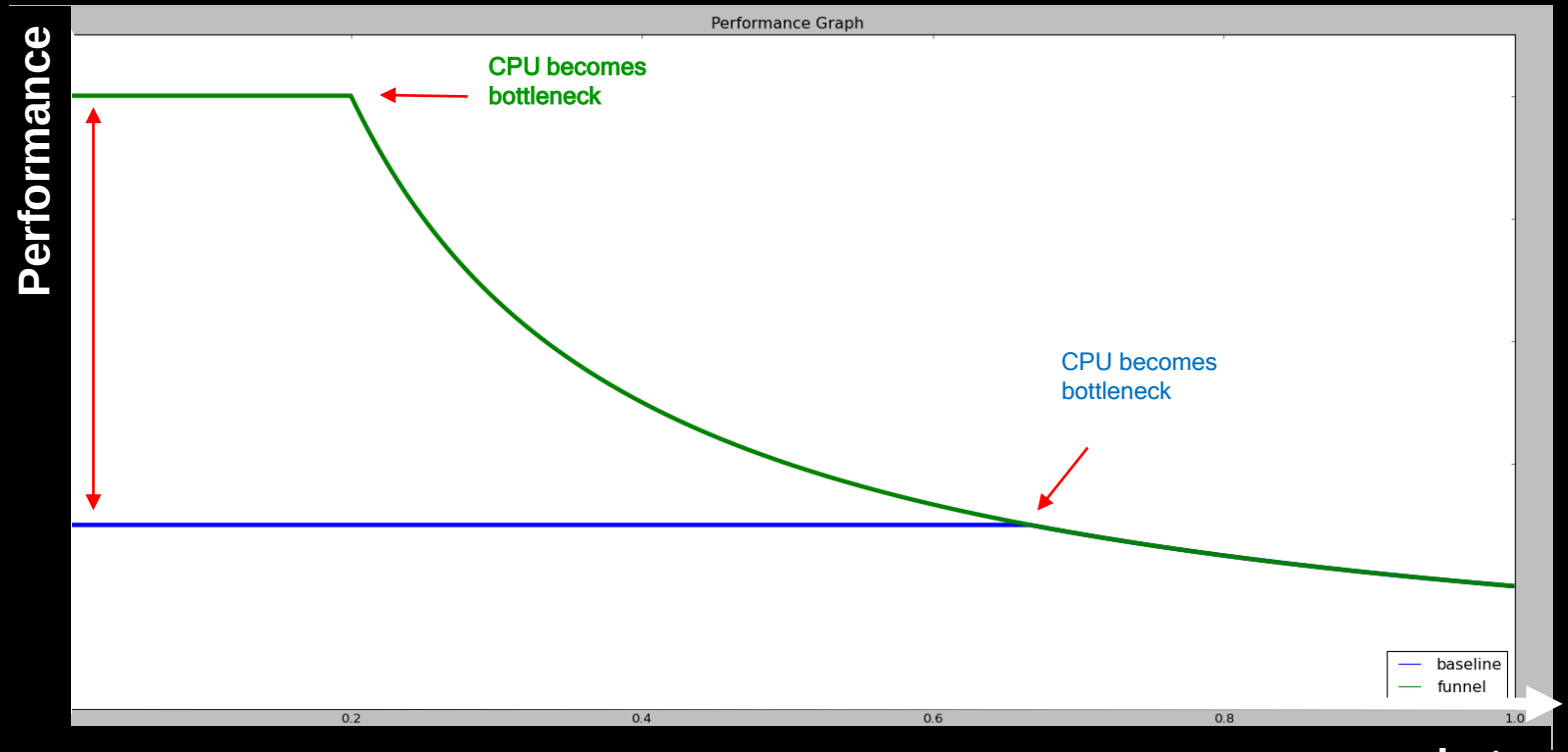
Funnel Configurations



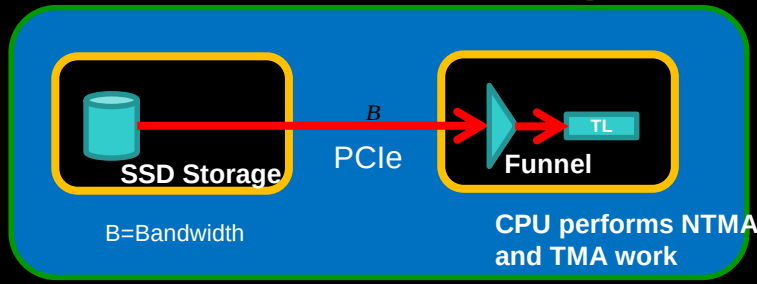
Funnel Performance

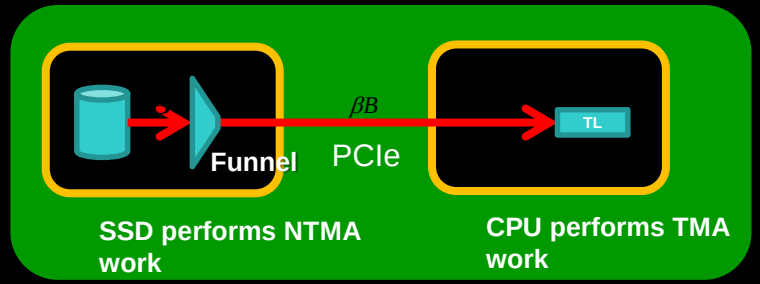


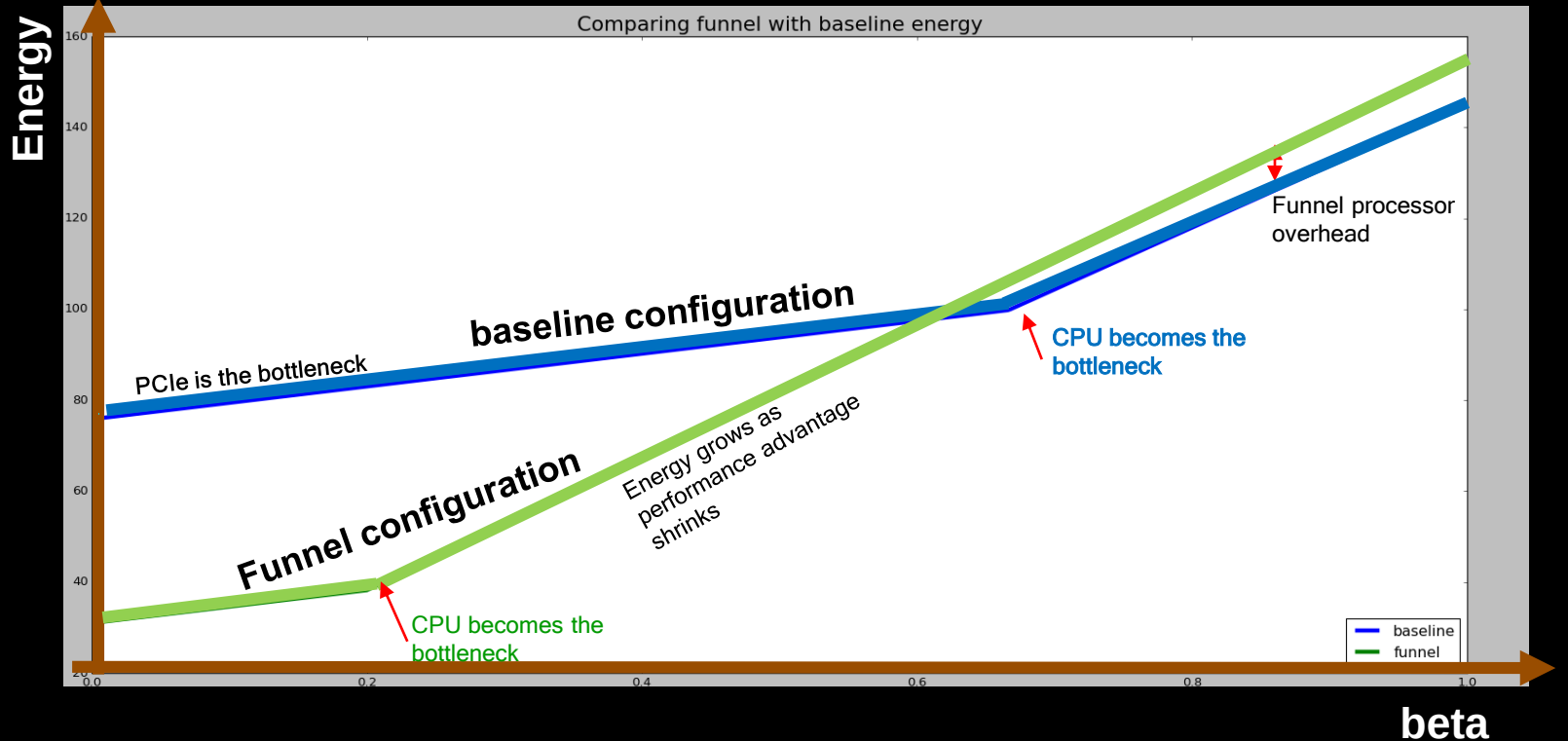




Funnel energy







Solution: ?

- Non-Temporal Memory Accesses should be processed as close as possible to the data source
- Data that exhibit Temporal Locality should use current Memory Hierarchy
- Use Machine Learning (context aware*) to distinguish between the two phases
- Open questions:
  - SW model
  - Shared Data
  - HW implementation
  - Computational requirement at the “Funnel”

*Reference: “Semantic locality and Context based prefetching” Peled, Mannor, Weiser, Etsion in ISCA 2015

Summary

- Lots of potential icebreaking potential research
  - Hetro
  - Big Data related
    - Memory access is a critical path in computing
- Funnel should be used for:
  - Reduction of Data movement
  - Free up system's memory resources (re-Spark)
  - Solve the System's BW issues for "Read Once" cases
  - Simple-energy-efficient engines at the front end
  - Issues
    - ...


THANK YOU