Big vs. Small cores for Big Data.

4th workshop on Architecture and Systems for Big Data

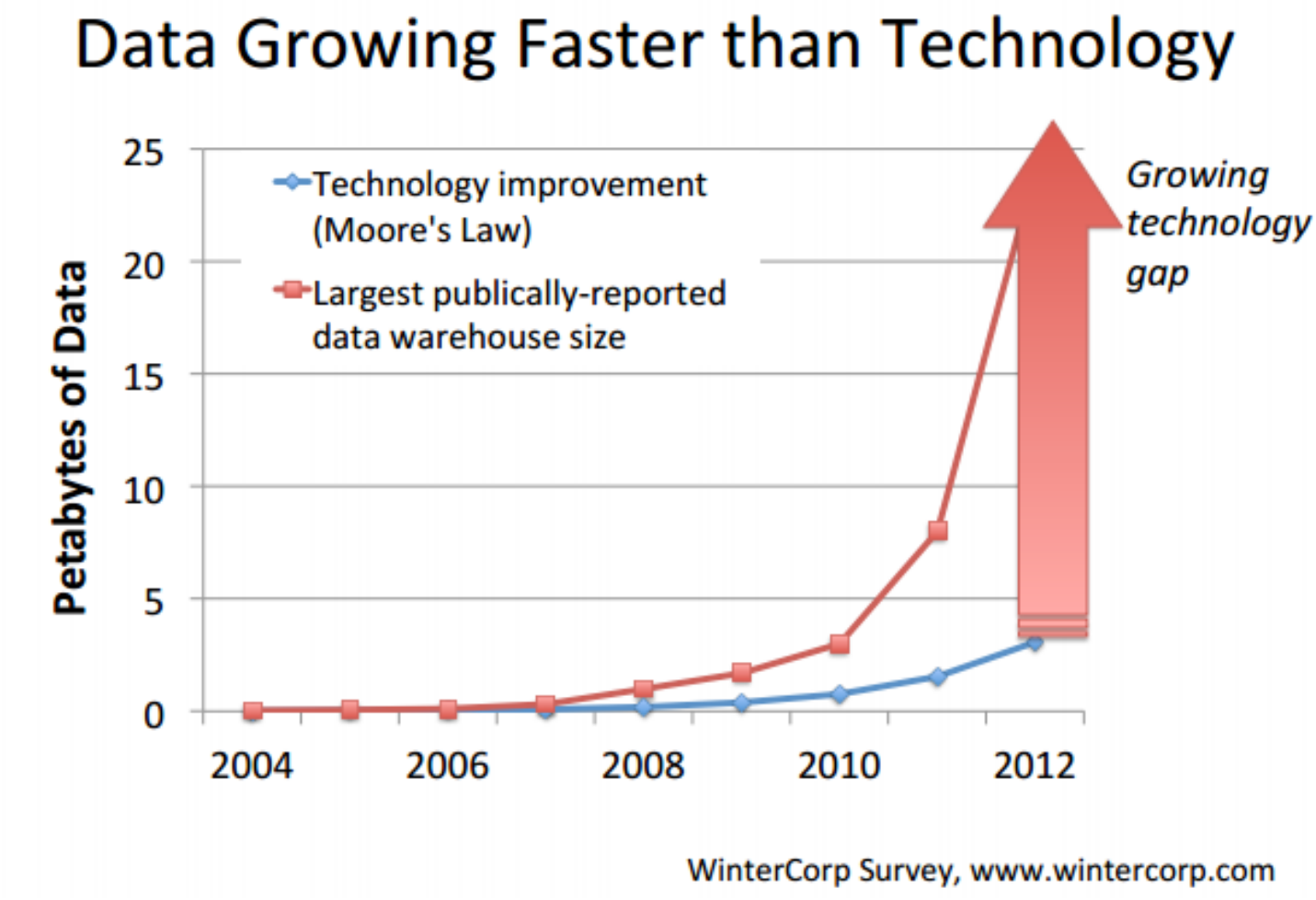
Prof. Avi Mendelson,
CS and EE departments, Technion
avi.mendelson@technion.ac.il

June, 2014

Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

A picture worth 1000 words



Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

Motivation and trends in processor development

- Two of the many versions of Moore's Law
  - Number of transistors on a die doubles every 18 months (the original form)
  - Measured performance of computer systems doubles every two years (one of many variations)
- Implications:
  - The same software model was used for different system generations
  - Allow predictability of performance and capabilities
  - Allows to maintain prices and revenue for both HW and SW based companies.

Process Technologies – the right turn:

- **New process could not achieve "ideal shrink" anymore**
  - Still doubles transistor density
  - But with "less than ideal" speed improvement and with power cost.
- **Leakage is becoming a big issue**
- **Today: it gets worse:**
  - Vt scaling, Variability and leakage are BIG issues

Performance is limited by power, energy and thermal

HW solution -- “go parallel”

- A simple power calculation of active power is:
  - Active power: Power = αCV^2f
    (α : activity, C: capacitance, V: voltage, f: frequency)
    Static power is out of the scope of this model.

- Since voltage and frequency depend on each other (between V_{max} and V_{min}), approximate power change in respect to freq. change as:

  $$\Delta Power \sim (\Delta f)^{2.5}$$

  (in theory it should be a factor of 3, in reality the factor is closer to 2.5)

- A naïve tradeoff analysis (assuming frequency maps to performance)
  - Doubling performance by increasing frequency grows power exponentially
  - Doubling performance by adding a core, grows power linearly

- Conclusions:
  (1) As long as enough parallelism exists, it is always more power efficient to double the number of cores rather than the frequency in order to achieve the same performance.
  (2) In thermally limited environment POWER == PERFORMANCE

How many cores we need?

- In order to maintain the “Moore law”, we expect to double the number of cores (performance) every generation.
- Current computer processors’ road maps are divided between
  - Multicore – small number of “big” cores, each of them maintains single-threaded performance – e.g, 1,2,4,8,16…
  - Manycore – large number of small cores, each of them shows reduced single threaded performance – e.g., 64, 128, 256, 512, 1024 ….

Deja-vu -- we have been there before

During the late 80's – 90's

Multi-cores:

- Intel Paragon
- Meiko
- SGI
- IBM SP1, SP2,…
- Multi More

Many cores:

- CM1-CM4
- Vector machines
- iWRAP –systolic arrays
- Transputers
- Many more

Too many companies went bankrupt because of these ideas.

Root cause → it's was mainly due to Software related issues

AND… now comes the Dark Silicon era

From "Dark silicon and the end of multicore scaling", Esmaeilzadeh, H., et al, ISCA 2011

- Even at 22 nm, 21% of a fixed-size chip must be powered off, and at 8 nm, this number grows to more than 50%..

ecocloud

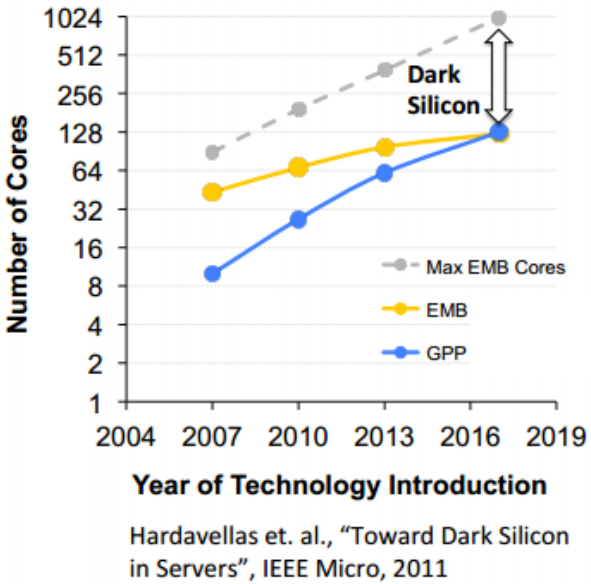
Dark Silicon: End of Multicore Scaling

Can not power up chip for fully parallel SW

Parallelism has limits even in Servers!

Must:

- specialize
- selectively power up



Hardavellas et. al., "Toward Dark Silicon in Servers", IEEE Micro, 2011

Dark Silicon will limit the number of cores we can simultaneously operate on a die.
How it will effect our future systems?

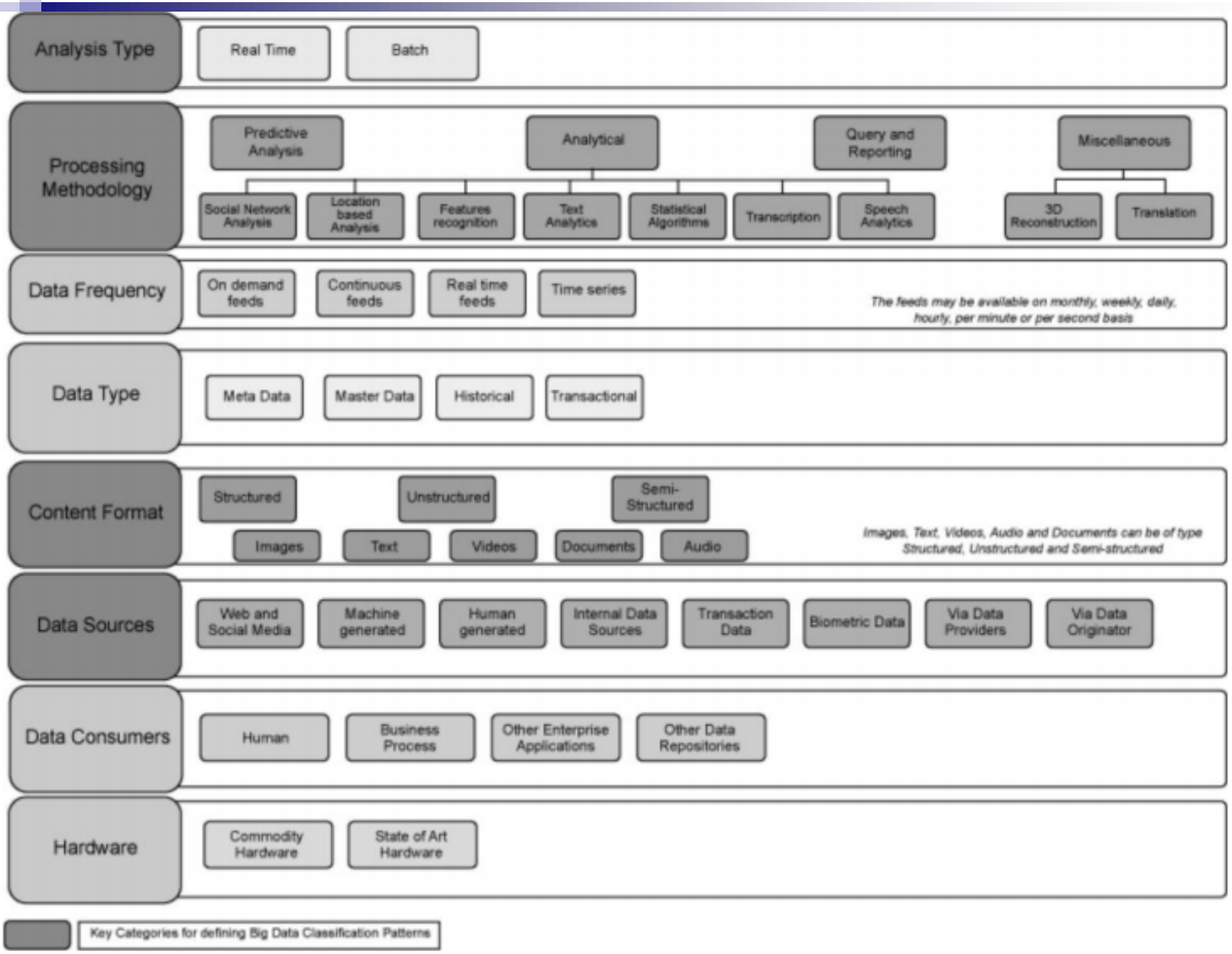
Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

What is “big data”

- Data is growing exponentially
  - it is expected that the size of “stored digital data” in the world will reach 35 Zettabytes until 2020.
  - We already have single files of size of Petabytes each
- Big data is not only about storage, but also about creating new usage models and new capabilities; i.e.,
  - Continuous tracking of massive number of sensors to improve health, quality of life, machines, etc.
- There are many types of “big data”, each has different requirements and characteristics

SOURCE "IBM"


Analysis Type
Real Time
Batch
Processing Methodology
Predictive Analysis
Analytical
Query and Reporting
Miscellaneous
Social Network Analysis
Location based Analysis
Features recognition
Text Analytics
Statistical Algorithms
Transcription
Speech Analytics
3D Reconstruction
Translation
Data Frequency
On demand feeds
Continuous feeds
Real time feeds
Time series
The feeds may be available on monthly, weekly, daily, hourly, per minute or per second basis
Data Type
Meta Data
Master Data
Historical
Transactional
Content Format
Structured
Unstructured
Semi-Structured
Images
Text
Videos
Documents
Audio
Images, Text, Videos, Audio and Documents can be of type Structured, Unstructured and Semi-structured
Data Sources
Web and Social Media
Machine generated
Human generated
Internal Data Sources
Transaction Data
Biometric Data
Via Data Providers
Via Data Originator
Data Consumers
Human
Business Process
Other Enterprise Applications
Other Data Repositories
Hardware
Commodity Hardware
State of Art Hardware
Key Categories for defining Big Data Classification Patterns

Characterization of Big-Data workloads

- It is commonly agreed that “Big Data”
  - has limited locality, unless huge local memory is used.
    - *I/O and memory management are critical for many applications*
  - Massively parallel
    - Utilization of resources approximates performance.
- But
  - This is mainly true for the “map” part, the “reduce” behaves differently and on the performance critical path in many cases.
  - There are many applications that can take advantage of locality and efficient access to caches.
  - For on-line and real-time Big Data applications, compute power, and predictable computation time may be more important than utilization.

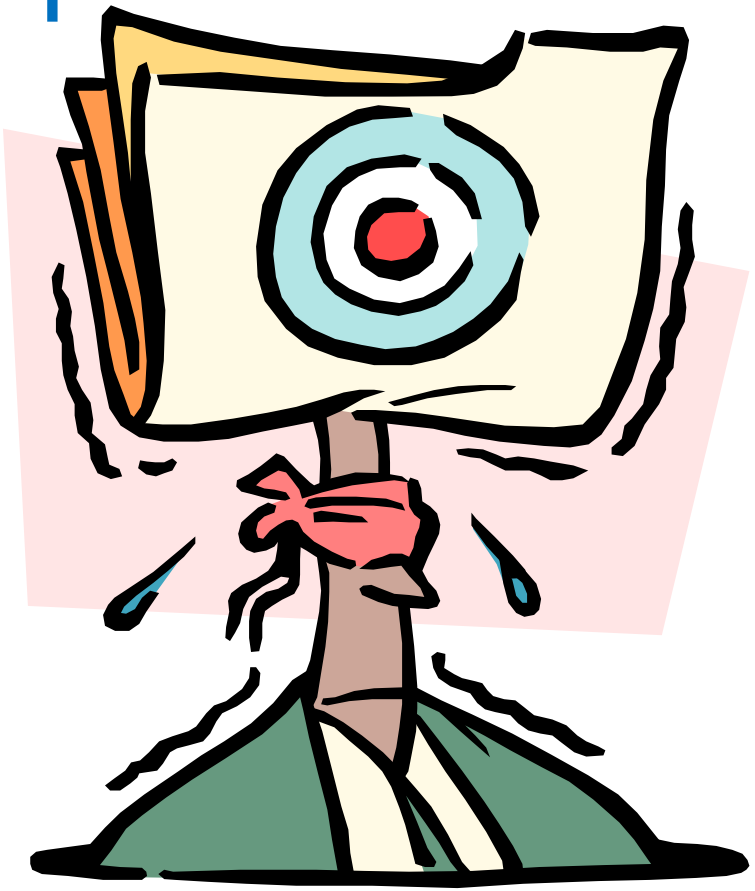
Research on Big Data

- This is a great area. You can get any result you like by "carefully choose" your parameters ☺
- Two Examples (I have many more)
  - Impact of TLB
    - There are quite a few research that indicate that TLB and page walk are critical for Hadoop applications such as Analytics (form Cloud-Suite, EPFL).
    - My student repeat the experiment, using Intel machines and found that TLB has negligible impact on the **same** benchmark
    - We use different physical memory sizes
  - Impact of JVM
    - Use different JVM and according to our experiments, you can gain (or loose) up to 40% overall performance, and different efficiency breakdown

Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

Big cores or Small cores – is this the right question?



Big cores or Small cores for Big Data?.

- Thermal and Energy consumption are the main criteria (Energy = power over time)
  - but not the only one; e.g., response time and predicted performance are very important for many applications
- The “obvious” answer is:
  - For batch processing that has enough parallelism, many (small and efficient) cores are better
  - For On-line processing and for activities on the “Critical path”, multi (big) cores are preferred.
- Does the bigLITTLE model presented by ARM can satisfy both environments – only partially

Second look at “batch processing”

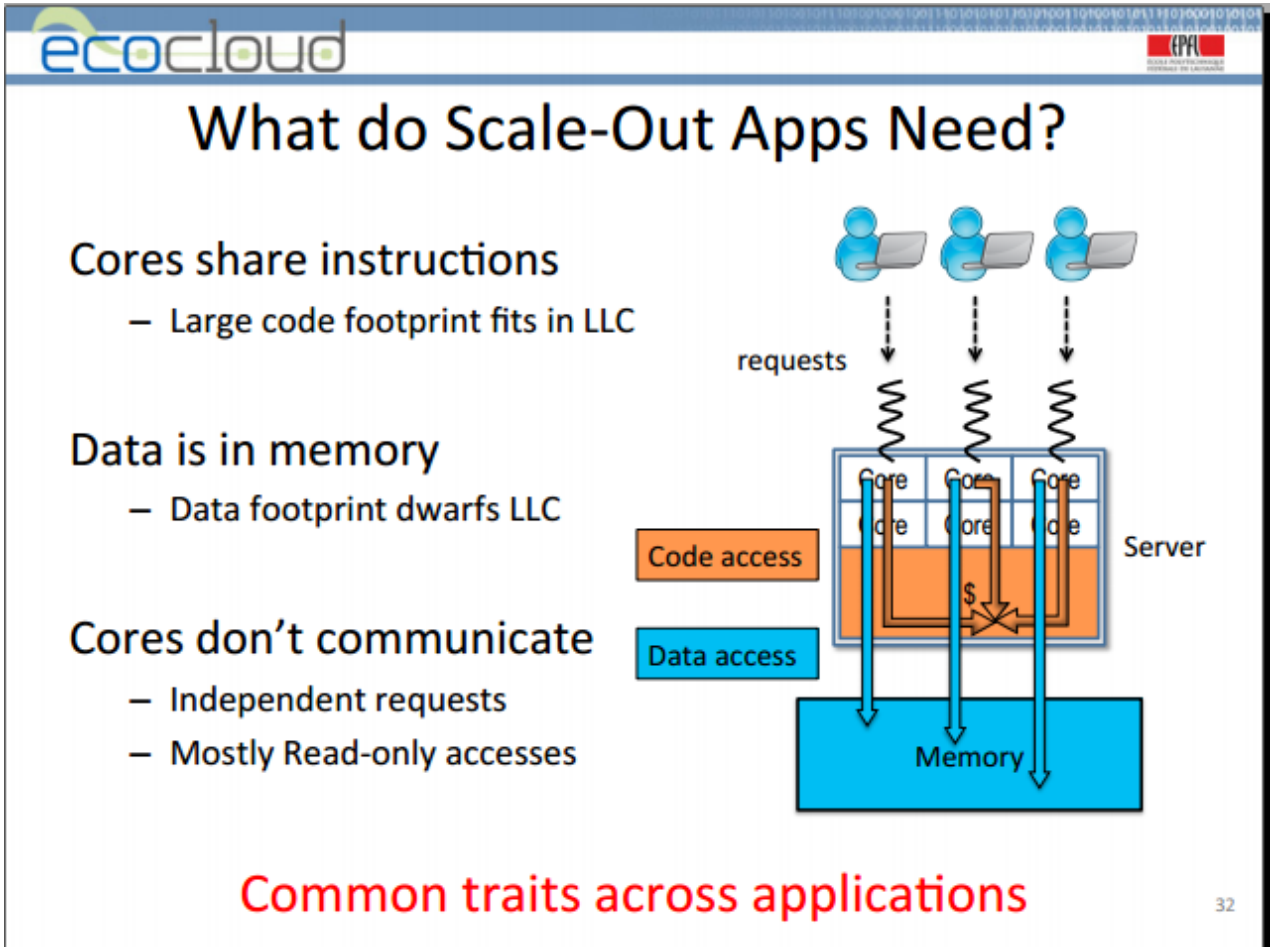
- Although SW may looks like having massive number of independent threads increasing number of cores come with a cost
  - increasing the number of cores, increases also the “sequential part of the code”; e.g., cost of synchronization, and so reduce the utilization of the system
  - Pressure on the caches
  - Pressure on I/O and memory access
- Big cores have better I/O and bus systems and can better utilize latency via OOO mechanisms

Other alternative – 1 Heterogeneous computing

- Integrate different types of processing units into the same die (or part of the same system).
- Different HW parts are optimized to handle different types of workloads; e.g.,
  - Many cores (GPU) are optimized for massive parallel processing
  - Big Cores are optimized for memory latency sensitive applications.
- The best of all worlds – If the software can efficiently use it.

Other alternative 2 – dedicated processor EPFL

proposal for “scale-out architectures



What about I/O and memory

- This is out of the scope of my talk but
  - Increasing the number of cores increase the pressure on the I/O.
  - RDMA is a great direction, but it is not sufficient. New generation of RDMA may be needed.
- 3D stacking is must and is happening.
  - Does it change the way we will build systems? I assume it will, but “out of the box” thinking is required
- Need to re-architect the memory subsystems to avoid TLB-shootdowns and related side effects
  - Need to integrate it with RDMA

Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

Does Moore's law dies ?

- More law, as was defined by Moore is still live
  - Process technology keep doubling the number of transistors on die, but
    - Less frequent
    - Very high cost
- The spirit of Moore law is not always kept
  - For specific applications, such as massive parallel, the effective performance continue to grow exponentially.
  - For general applications, the performance improves very slowly mainly due to need to change the SW in order to take advantage of new HW capabilities.

Domain specific solutions are the future

- Domain specific can use massive parallel processors
  - Application/HW/SW co-design is essential for getting good results.
  - When right SW/HW interfaces are defined; e.g., CUDA, future optimizations of SW and HW are aligned and so can keep growing exponentially and fulfill Moore's law
  - Applying the same techniques to different domains, not always provide the expected results in terms of power and performance.
- Do we need DSL (domain specific languages) for that?
  - DSL can survive in selective communities, such as programming FPGA with Verilog.
  - I believe that languages such as CUDA and OpenCL will not survive and in the future will become a derivative of C++ (such as in C++AMPS) or Java (or similar, such as Phyton or C#)

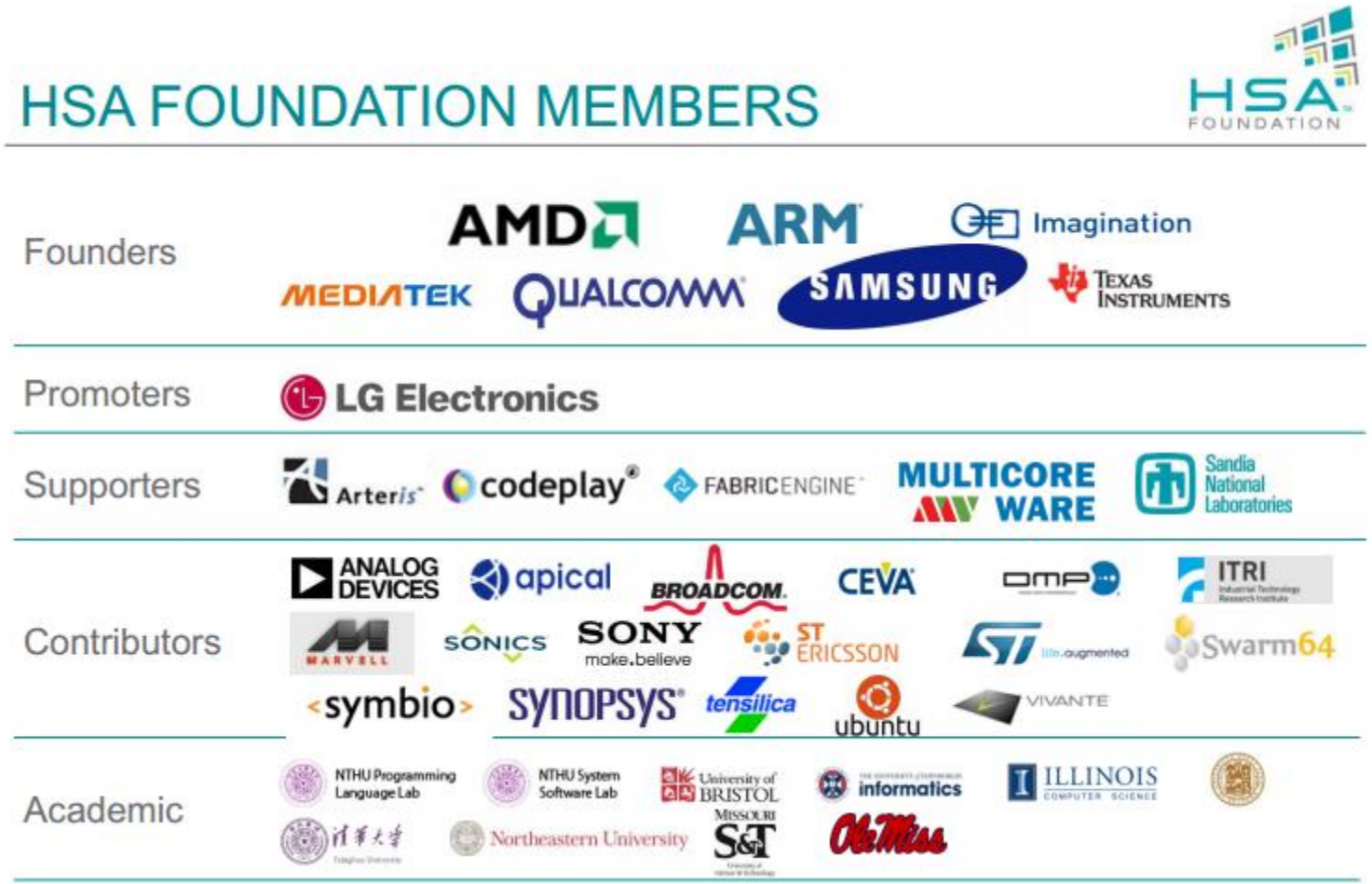
Do we need Domain Specific Hardware?

- For the time being we may need special purpose Hardwar for Domain Specific environment; e.g.,
  - Optimized system for “batch-MAP”
  - Optimized system for “reduce”
- DVFS and Turbo helps a lot to mitigate the gap between “small” and “big” cores
- At the SoC level we can integrate different types of architectures such as GPGPU or integrating DSP and/or FPGA can be very efficient for specific domains
- In the future, we will take advantage of **Dark Silicon**; e.g., build heterogeneous systems where different subsystem are used for at a given time

Can we program such a system?

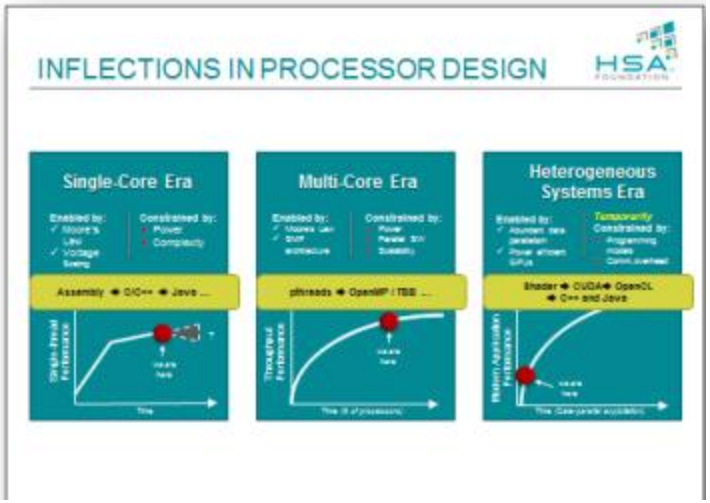
- The key is to build **new** SW/HW interfaces that will allow the SW to take maximum advantage of the HW, using Domain specific characterizations.
- We need to move the control over the Hardware from the OS to the user; i.e., to allow user mode code to control internal HW mechanisms and HW to expose internal behaviors such as power. thermal, utilization to the SW in a better way than the current performance counters.
- We need an OS that will be aware of the Domain it supports.

HSA is an important direction



WHAT IS HSA ALL ABOUT ?

- "Bring Accelerators forward as a first class processor"
  - Unified address space, pageable memory, coherency
  - Eliminate drivers from dispatch path (user mode queues)
- Standardized SW stack built on top of a set of HW requirements
  - Improve interoperability between IP vendors
- Unified Architecture for Accelerators
  - Start from GPU, extend to DSP / FPGA / Fixed-Function Acc , etc.
- SoC Centric
  - Major features are optimal for SoC environment (same memory/die)
  - Support of distributed system is possible, yet inefficient (PCI atomics, others)



Source: Ofer Rosenberg talk at "TCE system day", Technion, June, 2014

HSA WORKING GROUPS

- HSA Systems Architecture
  - hUMA – Unified Memory Model
  - hQ – HSA Queuing Model
- HSA Programmer Reference Specification
  - HSAIL – HSA Intermediate Language
- HSA System Runtime
- HSA Compliance
- HSA Tools

http://hsafoundation.com/standards/ 9

Source: Ofer Rosenberg talk at “TCE system day”, Technion, June, 2014

Agenda

- Background
  - Multi/Many/Big/Little/dark Silicon
  - Big Data Characteristics
- Put it all together
- Future directions (my personal view)
- Conclusions and remarks

Take away

- Big Data is not only about data, it is about creating new usage models that use huge amount of data
- These new usage models have different requirements and so cannot be satisfied by a single compute model (HW and SW)
- Power and Thermal are the main issues (at least for now) and so every mechanism we propose needs to be judged in respect to that
- Heterogeneous systems seem to be the key for the solution, but new HW/SW interfaces are needed
  - At user level
  - At system Level

Issues we didn’t addressed

- This is just the beginning since we also need to handle
  - I/O
  - Memory
  - Resource management, including power management, thermal management and more
  - Security is a big issue
  - OS
- ETC.

Last but not least Research on Big Data

- We need to re-think on the way we perform research in this area
  - Simulators – most of them ignore I/O and “Big Data” characteristics
  - Workloads – too few and most of them “in-Memory”
  - Impact of architectural and system parameters on the results.

Thanks!