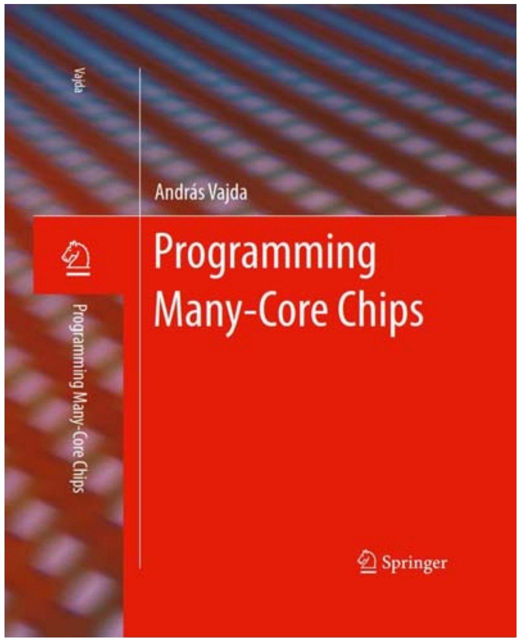
WHAT CLOUD COMPUTING CAN TEACH US ABOUT EMBEDDED MANY-CORE PROGRAMMING?

András Vajda

Expert, Cloud Computing, Ericsson

ABOUT ME

- Chief architect of several key telecom products @ Ericsson
  - Hundreds to thousands of processors, millions of lines of code
- Initiated and led Ericsson research activities in the area of many-core programming and cloud computing
- Author of the book “Programming Many-core Chips” (2011)
- Social stuff
  - Blog: http://www.a-vajda.eu/blog
  - Twitter: @andrasVajda



DISCLAIMER

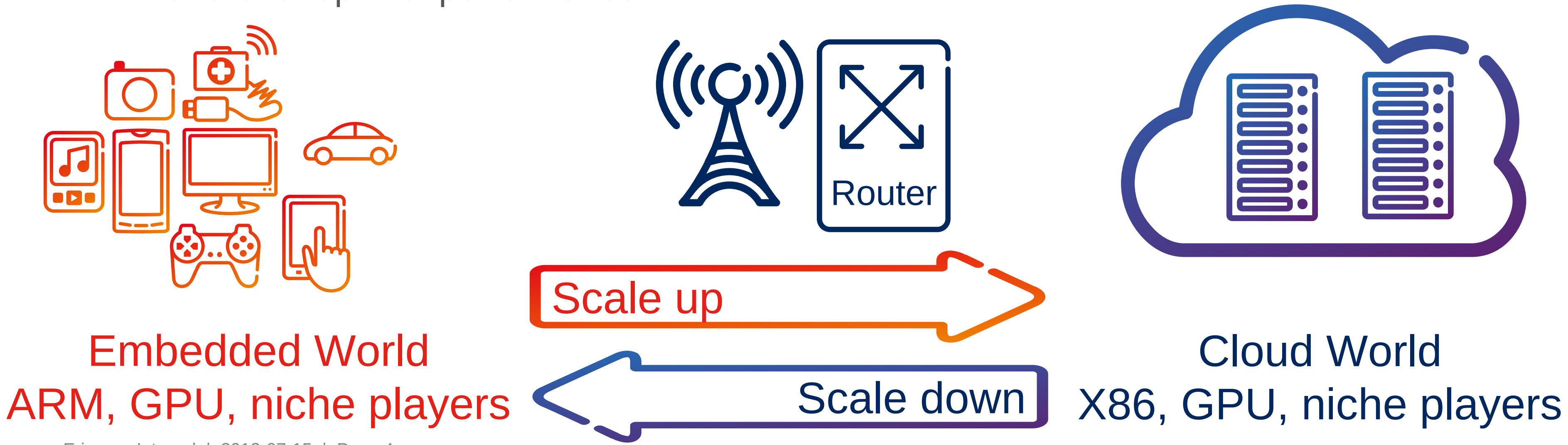
This presentation represents my own views and those probably don’t match with the views of my employer.

I will make some claims you will probably disagree with.

I’m known to have be wrong many times – I hope that will serve as an explanation/consolation/excuse ☺

THE WORLD IS (BECOMING) BINARY

- (Almost…) everything is soon either a connected embedded system or migrates to a centralized elastic execution platform (“cloud”)
  - With the notable exception of networking infrastructure
- The number of widespread compute platforms is shrinking constantly
  - “… and there will be two” (or maybe three)
  - Applies to networking, too
- We are moving to a more and more homogeneous world
  - The need to address software complexity and HW design cost will get the upper hand over optimal performance



A SMALL DICTIONARY

Data Center		Many-core chip
Server		Core
Server memory		Cache
Storage	Similar to	Memory
DC Network fabric		On-chip interconnect
Data Center		Many-core chip
Cloud		Multi-processor system

CLAIMS FROM THE INDUSTRY

- Consumed total energy for a task will be more important than power
  - I.e., it’s ok to burn more power if the result is achieved in a shorter time
- Compute and storage will be free (or several orders of magnitude cheaper) compared to the cost of moving bits around (and keeping it consistent)
  - Volume of data is growing faster than the technology for moving it around
- Cost of software development makes suboptimal performance the new normal
  - but it’s ok to waste some capacity, right?
- Multi/Many-core chips will eventually mandate the use of virtualization solutions
  - Best solution so far for preserving legacy

LESSON NO 1: EMBRACE VIRTUALIZATION

- Virtualization is so far the best technology for consolidating workloads without change of the software
  - All major ISAs embrace virtualization support

- Embedded systems have similar requirements as “DC-enabled” workloads, with the addition of
  1. Support for real-time workloads and operating systems
     - Requires closer inter-working between RTOS and hypervisors
  2. Requirement on smaller footprint
  3. Support for “esoteric” operating systems

- It’s a reality for telecom systems, game consoles, mobile phones etc
  - Virtualization is being embraced by e.g. GPUs too

LESSON NO 2: MOVE COMPUTATIONS, NOT DATA

- If shuffling data around is expensive (and impacts consistency in a parallel system)…
  - …it's better to move the computation where the data is
- In a data center:
  - Detect when data stored elsewhere is being accessed
  - Start/move the VM where the data is stored
  - Schedule for exclusive execution (serialize access to data)
- In an embedded system:
  - lock the data to the cache of a core, shift execution to that core when needed
  - Bonus: you can skip cache coherence altogether and save power

WHY?

- Current shared memory models rely on cache coherence
  - Bringing in memory blocks to where the code needs it
  - Complex, high cost of design & verification

- Cache coherence only solves the easier part of the problem
  - Programmers still need to design and use synchronization mechanisms in order to design semantically correct software
  - This effort by programmer is not exploited by hardware in any meaningful way

- So...
  - ... Let the software define how caches are used and safe shared memory semantics are obtained

BASIC HARDWARE ORGANIZATION

Data Center	Many-core chip
Each compute node has storage it controls	Each core has local associated cache
No consistency synchronization between compute nodes	No cache coherence between distributed caches
All-to-all interconnect & ability to move computations	All-to-all interconnect & ability to move computations

SHARED MEMORY WITH NO COHERENCE

- The programmer uses synchronization mechanisms for semantically correct execution
  - Programmers today think in terms of ”sharing transactions” that need exclusive access to some parts of the memory
  - Locks, mutexes: acquisition and release of lock mark a transaction
  - Transactional memory: begin/end transaction mark a transaction
- These are hints about areas in the program that will access shared resources
- If we add information about *which shared resources* will be accessed ....
  - ... The compiler, run-time system and hardware have sufficient information to provide safe shared memory semantics

ACHIEVING SHARED MEMORY SEMANTICS

- Based on sharing transaction markings by the programmer ...
  - ... Compiler groups related code fragments into transaction groups (TG)
  - ... Each TG is assigned one core that will act as its guardian core
  - ... The execution of transactions always takes place on the corresponding guardian core
- Guardian cores ....
  - ... Maintain a (hardware) queue of tasks (transactions) to be executed
  - ... The queue enables (hardware) pre-fetching of data to cache – next transaction to execute is known as well as which areas it will access
- Nested transactions require dead-lock detection and resolution mechanisms between cores
  - Based on tracking nested blocking of guardian cores
  - Can be done in hardware

BENEFITS

- For the programmer...
  - no need to worry about synchronization and dead-lock, as long as all transactions are correctly marked
- For chip architecture...
  - no need for complicated cache coherence
  - Flexibility – hardware adapts to software
  - Deterministic serialization of concurrent access enables better cache management – almost ”perfect” cache hit ratio

LESSON NO 3:
GO AHEAD, WASTE RESOURCES

- If your goal is…
- … to complete a task in as short time as possible and at minimum energy level

- Then…
- … it's ok to waste some computations if it helps complete the task faster AND at lower energy level
  (after all, in cloud compute resources are cheap)

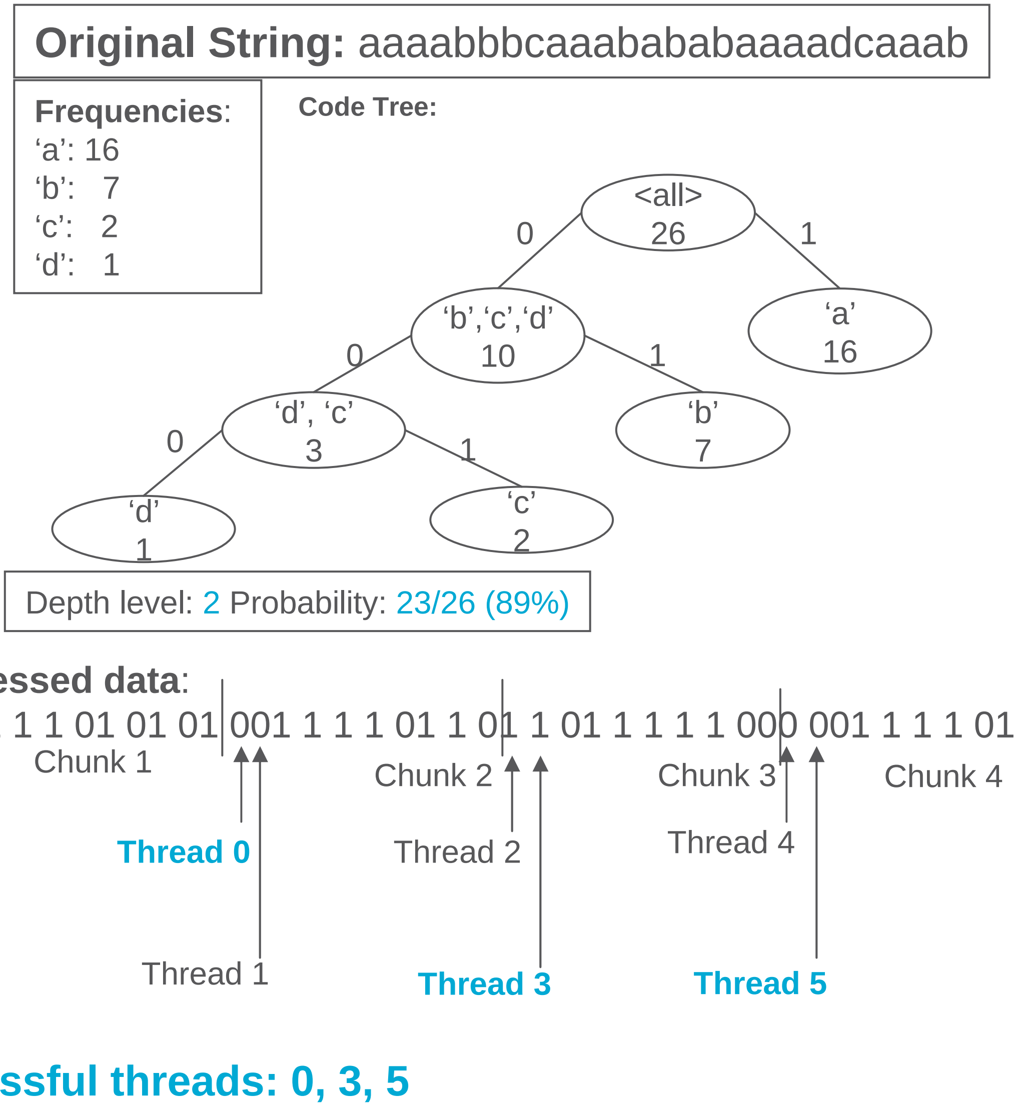
- What's the best way to waste it? ☺

SEMANTIC INFORMATION BASED SPECULATIVE EXECUTION

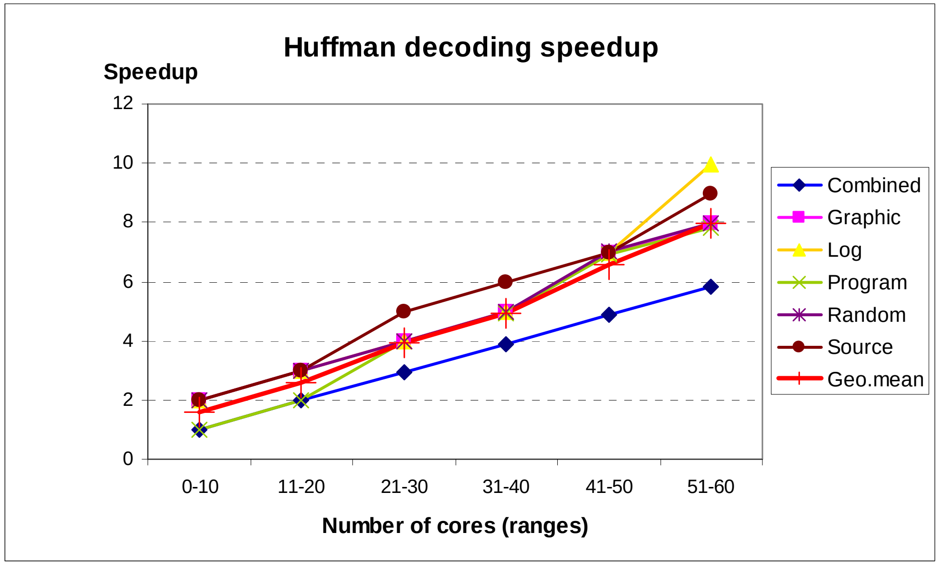
- Insight: even if an application has strong sequential dependencies, the type of input data may limit the space of possible execution paths
  - With a certain probability, one of these paths is the correct one
- With the right HW / run-time system support this can be exploited
- In a data center: execute the different execution paths in isolation, then pick the correct result

EXAMPLE: HUFFMAN DECODING

- Basic algorithm: assign shorter codes to more frequent symbols
- Decompression: hard to parallelize, as it's hard to deterministically identify where code boundaries are
- Idea: it's possible to calculate the probability ***P*** of having a code boundary within ***N*** consecutive bits
- Solution: apply data parallelism, but for each chunk, execute ***N*** parallel *speculative* version, each starting at different bit positions



PROBABILISTIC DATA PARALLELISM



- In Huffman decompression case:
  - Spatial Interval: ***N*** (in the example, 2)
  - Associated probability: ***P*** (in the example, 0.89)

  - Speedup on K cores for the example is

    S= (K-1) / 2 + 1,

    with probability 0.89

- Probabilistic data parallelism: data parallelism where chunk boundaries are defined as a ***spatial interval*** with ***associated probability***

- Probable speedup on K cores, with probability P:

  S = (K-1) / N + 1

- Speedup is dependent on the number of cores, the algorithm and the input data

- Speedup is achieved at constant energy consumption (using lower frequencies)

WHAT DID CLOUD TEACH US? (OR, RECONFIRMED WHAT WE KNEW)

- Compute resources are abundant; interconnect and data consistency is costly and complex
- Total energy is more important than power
- “infinite” resources and elastic “creation” of resources enable controlled and educated waste in order to address the first two concerns
  - Speculative execution, pre-computation etc
- Virtualization is a powerful technology for consolidation of workloads, uplift of legacy and elasticity

HOW WILL EMBEDDED SYSTEMS LOOK LIKE?

- … will be virtualized
- … will look more like “mini data centers”
- … will be more wasteful
- … will be orders of magnitude more complex (SW wise)
- … many will simply be extensions of a central data center
  - Enabled by the homogenization of compute environments & virtualization

THANK YOU!

ANDRAS.VAJDA@ERICSSON.COM

ERICSSON