C-DAC Four Days Technology Workshop

ON

Hybrid Computing – Co-Processors/Accelerators
Power-aware Computing – Performance of Applications Kernels

hyPACK-2013
(Mode-1:Multi-Core)

Lecture Topic :

Multi-Core Processors :
Algorithms & Applications Overview - Part-II

Venue : CMSD, UoHYD ; Date : October 15-18, 2013

Killer Apps of Tomorrow

Application-Driven Architectures: Analyzing Tera-Scale Workloads

- Workload convergence
  - The basic algorithms shared by these high-end workloads
- Platform implications
  - How workload analysis guides future architectures
- Programmer productivity
  - Optimized architectures will ease the development of software
- Call to Action
  - Benchmark suites in critical need for redress

Emerging "Killer Apps of Tomorrow" Parallel Algorithms Design

- Static and Load Balancing
  - Mapping for load balancing
  - Minimizing Interaction
  - Overheads in parallel algorithms design
- Data Sharing Overheads

Emerging "Killer Apps of Tomorrow" Parallel Algorithms Design (Contd…)

General Techniques for Choosing Right Parallel Algorithm

- Maximize data locality
- Minimize volume of data
- Minimize frequency of Interactions
- Overlapping computations with interactions.
- Data replication
- Minimize construction and Hot spots.
- Use highly optimized collective interaction operations.
  - Collective data transfers and computations
- Maximize Concurrency.

Programming Aspects Examples

Implementation of Streaming Media Player on Multi-Core

- One decomposition of work using Multi-threads
- It consists of
  - A thread Monitoring a network port for arriving data,
  - A decompressor thread for decompressing packets
  - Generating frames in a video sequence
  - A rendering thread that displays frame at programmed intervals

Source : Reference : [4]

Programming Aspects Examples

Implementation of Streaming Media Player on Multi-Core

- The thread must communicate via shared buffers –
  - an in-buffer between the network and decompressor,
  - an out-buffer between the decompressor and renderer
- It consists of
  - Listen to port ……..Gather data from the network
  - Thread generates frames with random bytes (Random string of specific bytes)
  - Render threads pick-up frames & from the out-buffer and calls the display function
  - Implement using the Thread Condition Variables

Types of Parallelism

Types of Parallelism

- Data parallelism
- Task parallelism
- Combination of Data and Task parallelism
- Stream parallelism

Evolving towards model-based Computing

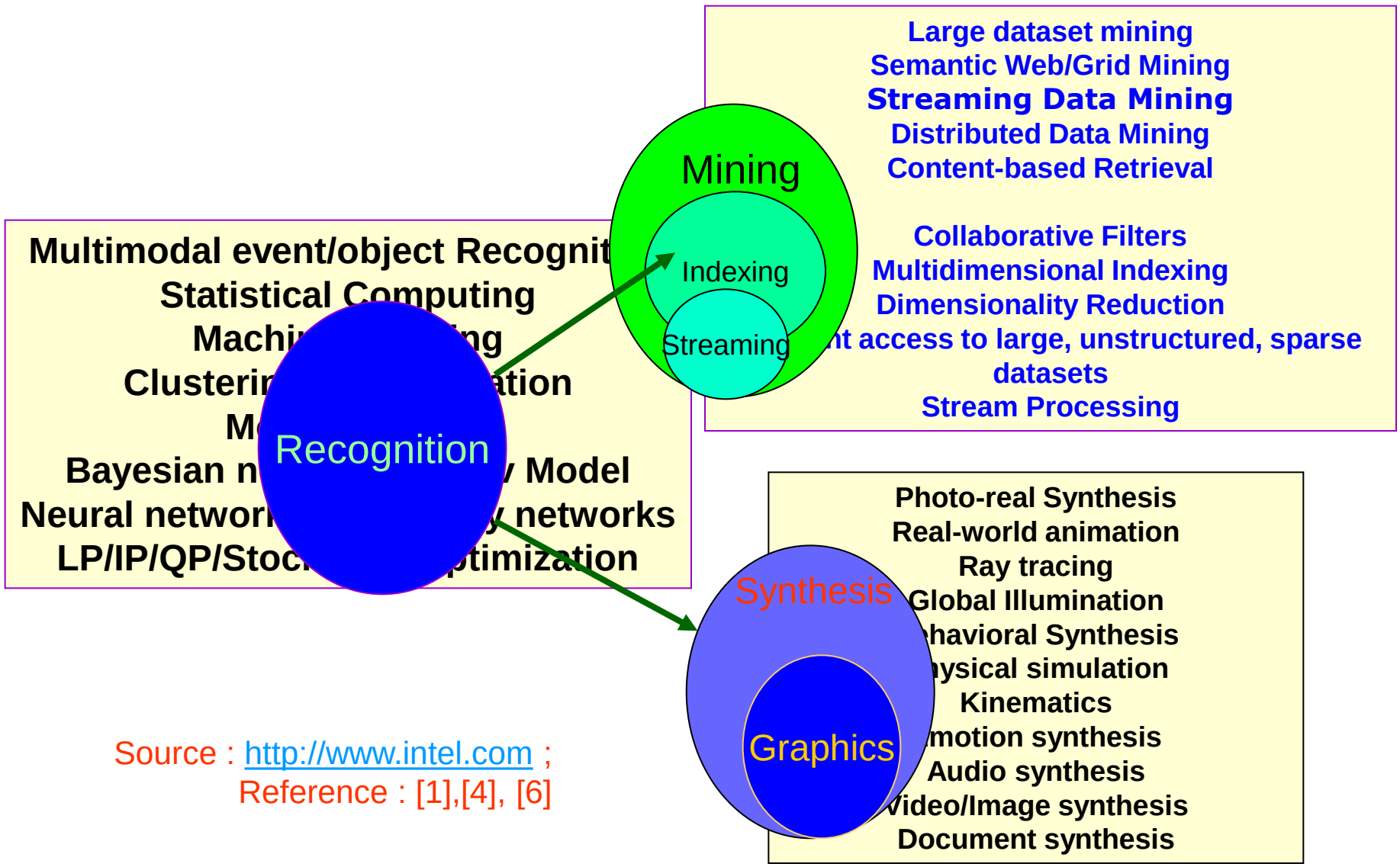
Large dataset mining
Semantic Web/Grid Mining
Streaming Data Mining
Distributed Data Mining
Content-based Retrieval

Collaborative Filters
Multidimensional Indexing
Dimensionality Reduction
Efficient access to large, unstructured, sparse datasets
Stream Processing

Multimodal event/object Recognition
Statistical Computing
Machine Learning
Clustering / Classification
Model-based:
Bayesian network/Markov Model
Neural network / Probability networks
LP/IP/QP/Stochastic Optimization

Photo-real Synthesis
Real-world animation
Ray tracing
Global Illumination
Behavioral Synthesis
Physical simulation
Kinematics
Emotion synthesis
Audio synthesis
Video/Image synthesis
Document synthesis

Evolving towards model-based Computing



Source : http://www.intel.com ;
Reference : [1],[4], [6]

Killer Apps of Tomorrow

- Workload convergence
  - The basic algorithms shared by these high-end workloads
- Platform implications
  - How workload analysis guides future architectures
- Programmer productivity
  - Optimized architectures will ease the development of software
- Call to Action
  - Benchmark suites in critical need for redress

Target For

Bigger *OR* Smaller Cores

Performance *OR* Scalability

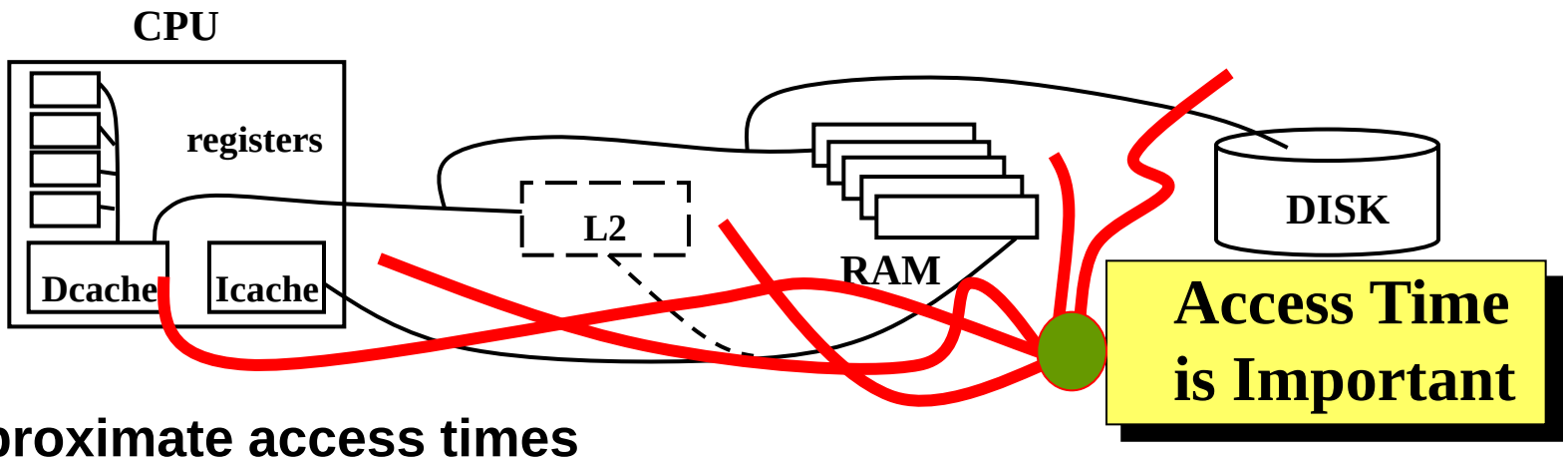
Compute *OR* I/O Intensive

Cache Friendly OR Memory Intensive

Source : http://www.intel.com ; Reference : [6]

The Memory sub-system : Access time

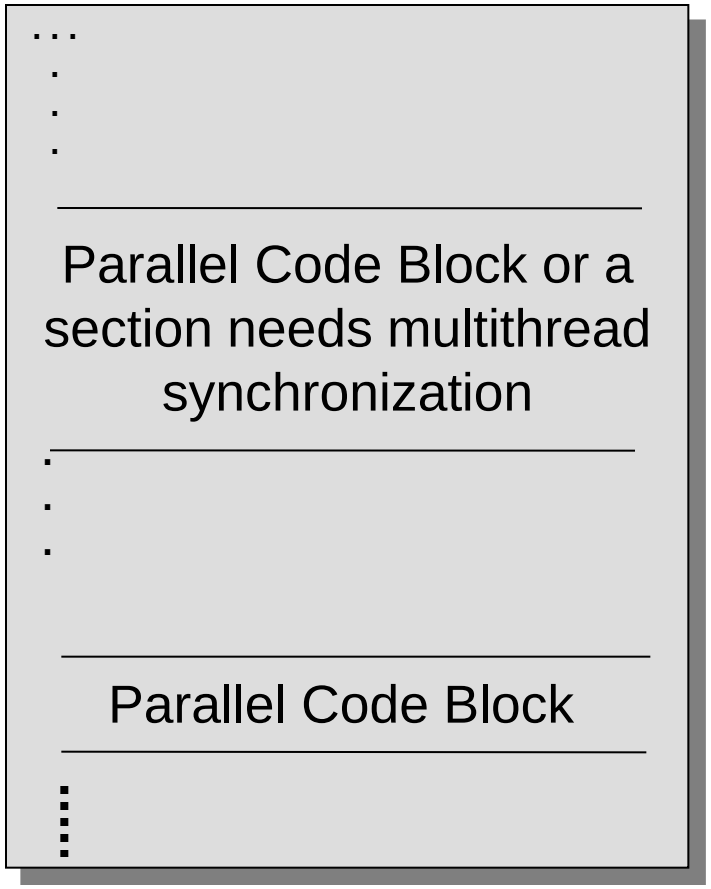
- A lot of time is spent accessing/storing data from/to memory. It is important to keep in mind the relative times for each memory types:



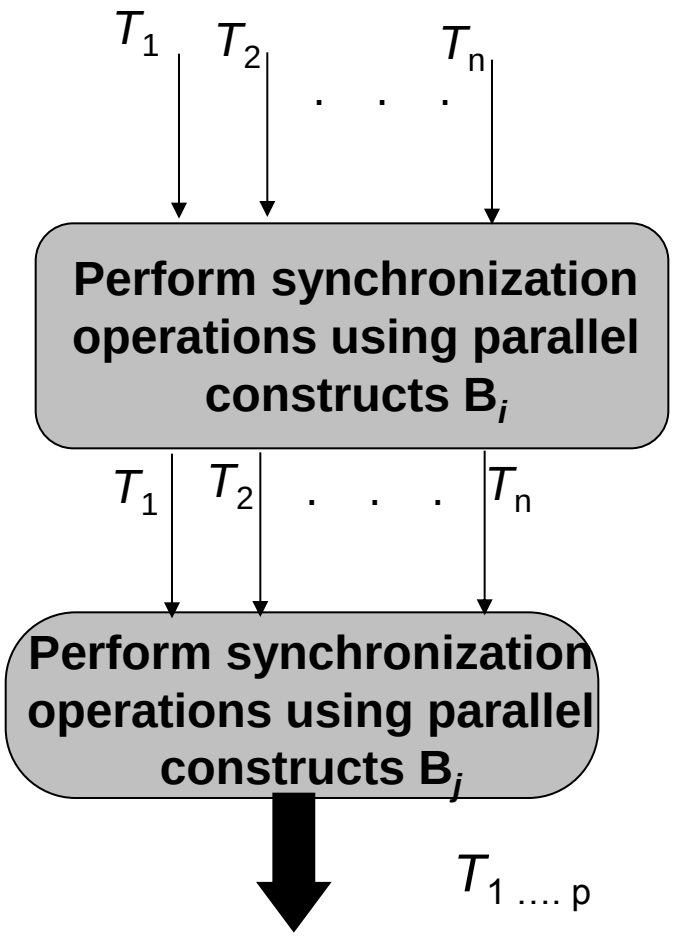
- **Approximate access times**
  - CPU-registers: 0 cycles (that's where the work is done!)
  - L_1 Cache: 1 cycle (Data and Instruction cache). Repeated access to a cache takes only 1 cycle
  - L_2 Cache (static RAM): 3-5 cycles?
  - Memory (DRAM): 10 cycles (Cache miss);
  - 30-60 cycles for Translation Lookaside Buffer (TLB) update
  - Disk: about 100,000 cycles!
  - connecting to other nodes - depending on network latency

Operational Flow of Threads for an Application

Implementation Source Code



Operational Flow of Threads



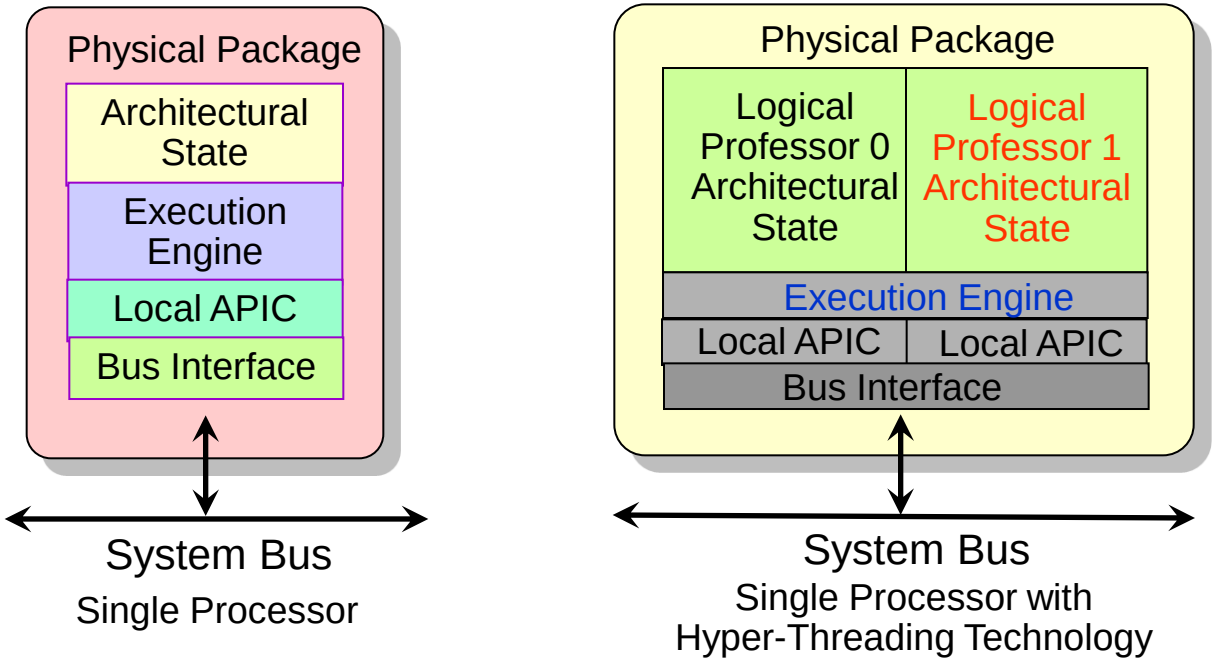
Source : http://www.intel.com ; Reference : [6]

Multi Core : Programming Issues

- Out of Order Execution
- Preemptive and Co-operative Multitasking
- SMP to the rescue
- Super threading with Multi threaded Processor
- Hyper threading the next step (Implementation)
- Multitasking
- Caching and SMT

Hyper-threading : Partitioned Resources

- Hyper-threading (HT) technology is a hardware mechanism where multiple independent hardware threads get to execute in a single cycle on a single super-scalar processor core.

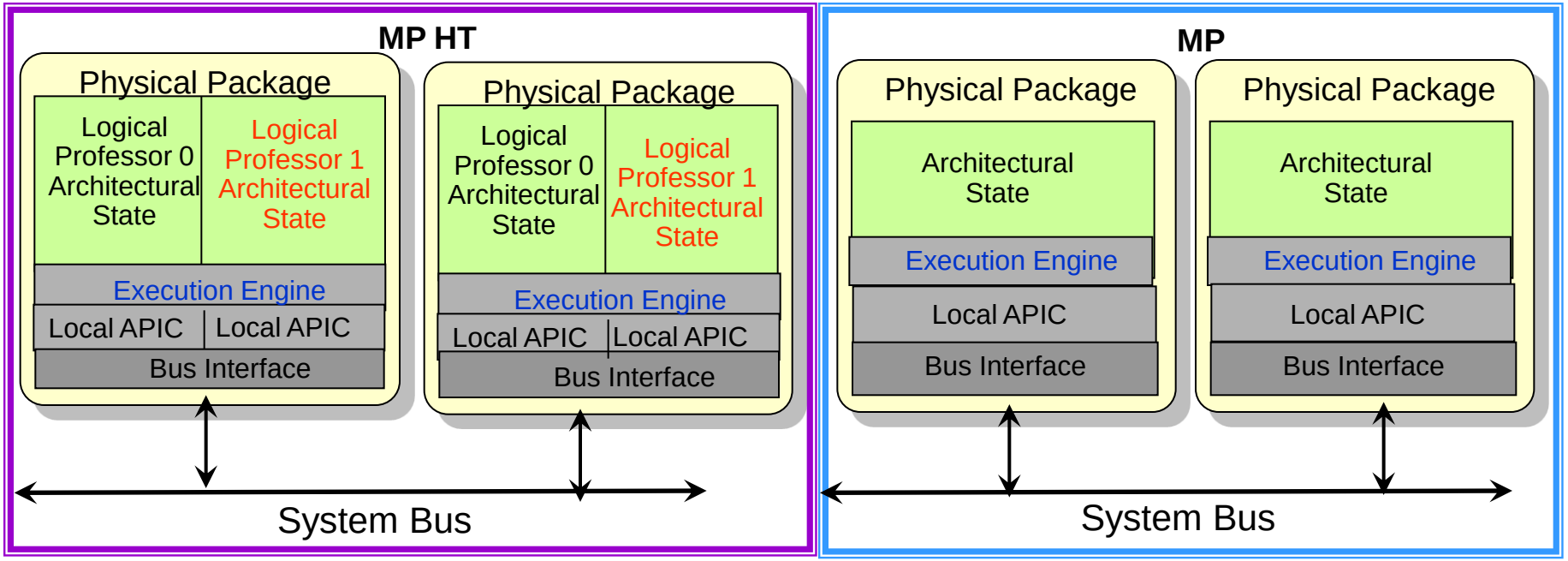


Single Processor System without Hyper-Threading Technology and Single Processor System with Hyper-Threading Technology

Source : http://www.intel.com ; Reference : [6], [10], [19], [23], [24],[29], [31]

Hyper-threading Technology

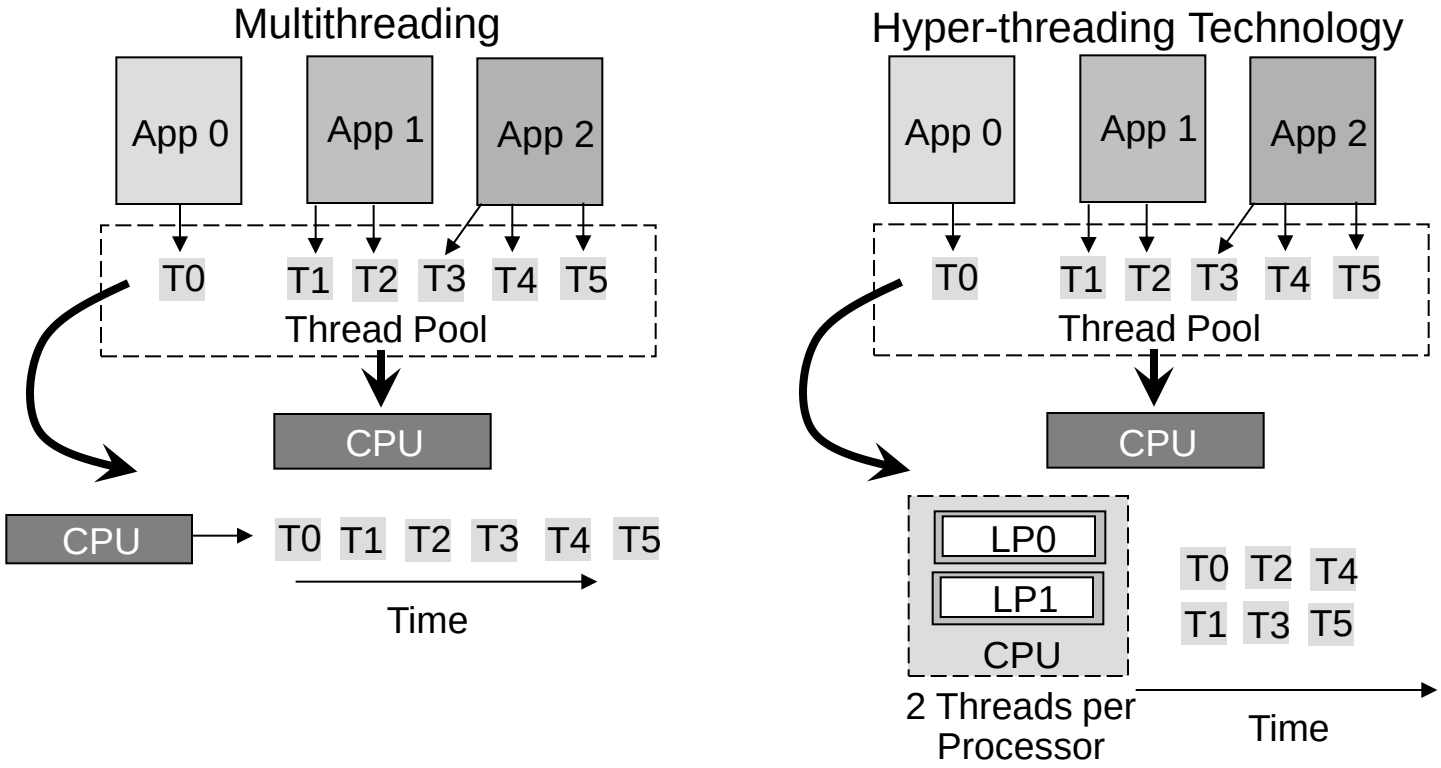
- Multi-processor with and without Hyper-threading (HT) technology



Source : http://www.intel.com ; Reference : [6], [10], [19], [23], [24],[29], [31]

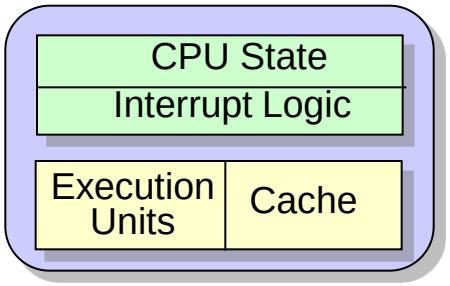
Multi-threaded Processing using Hyper-Threading Technology

- Time taken to process *n* threads on a single processor is significantly more than a single processor system with HT technology enabled.

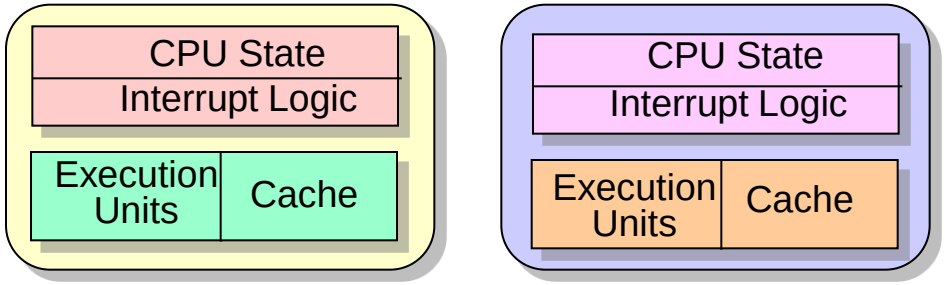


Source : http://www.intel.com ; Reference : [6], [29], [31]

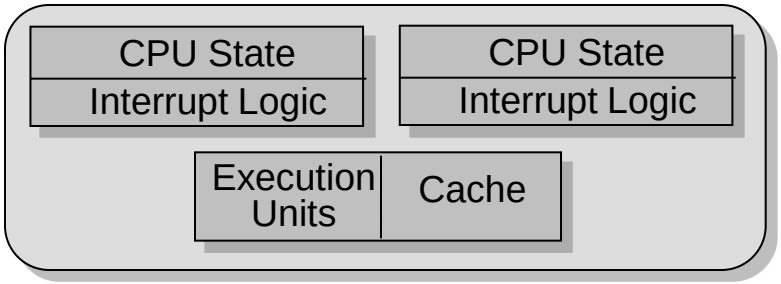
Simple Comparison of Single-core, Multi-processor, and multi-Core Architectures



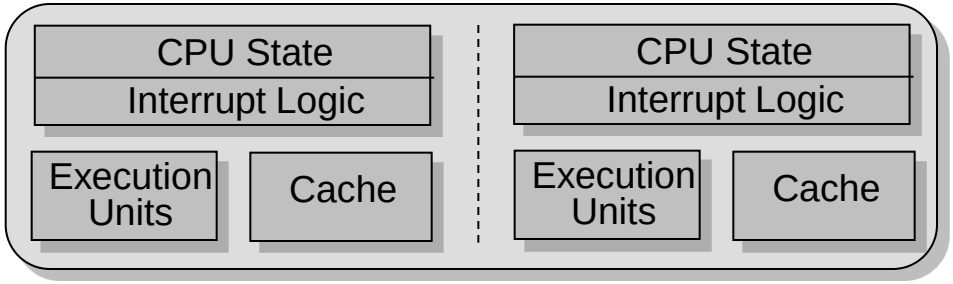
A) Single Core



b) Multi Processor



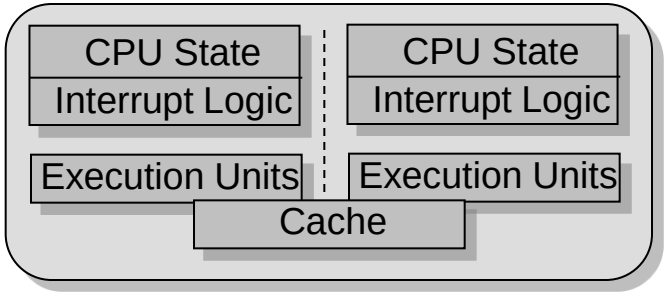
C) Hyper-Threading Technology



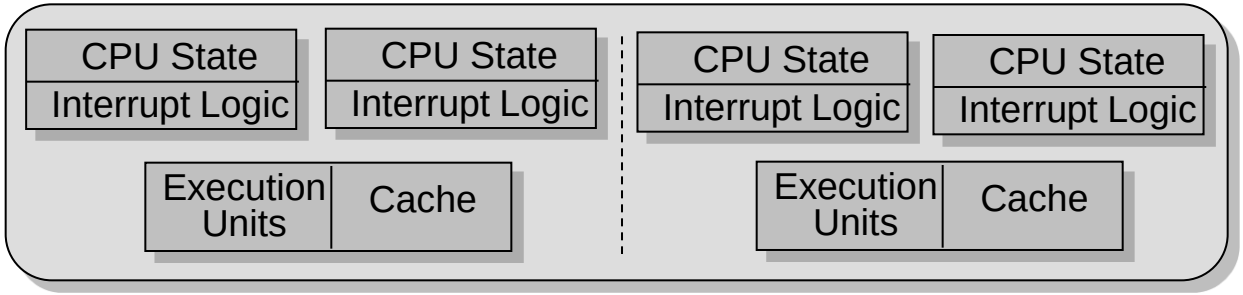
D) Multi-core

Source : http://www.intel.com ; Reference : [6], [29], [31]

Simple Comparison of Single-core, Multi-processor, and multi-Core Architectures



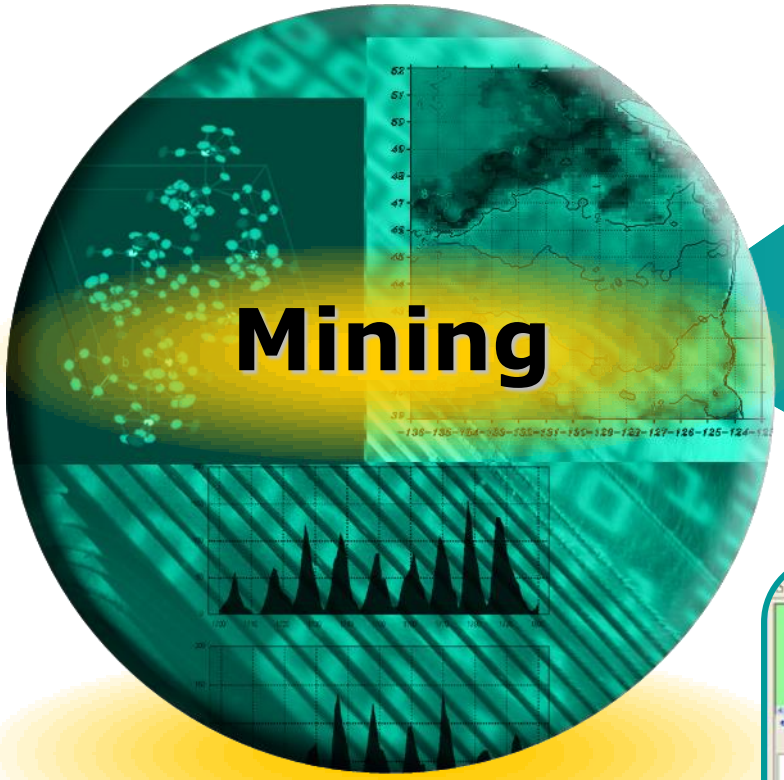
E) Multi-core with Shared Cache



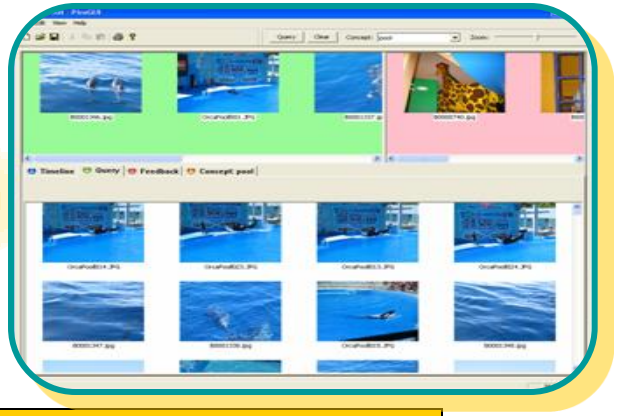
F) Multi-core with Hyper-threading Technology

Source : http://www.intel.com ; Reference : [6], [29], [31]

Emerging "Killer Apps of Tomorrow"



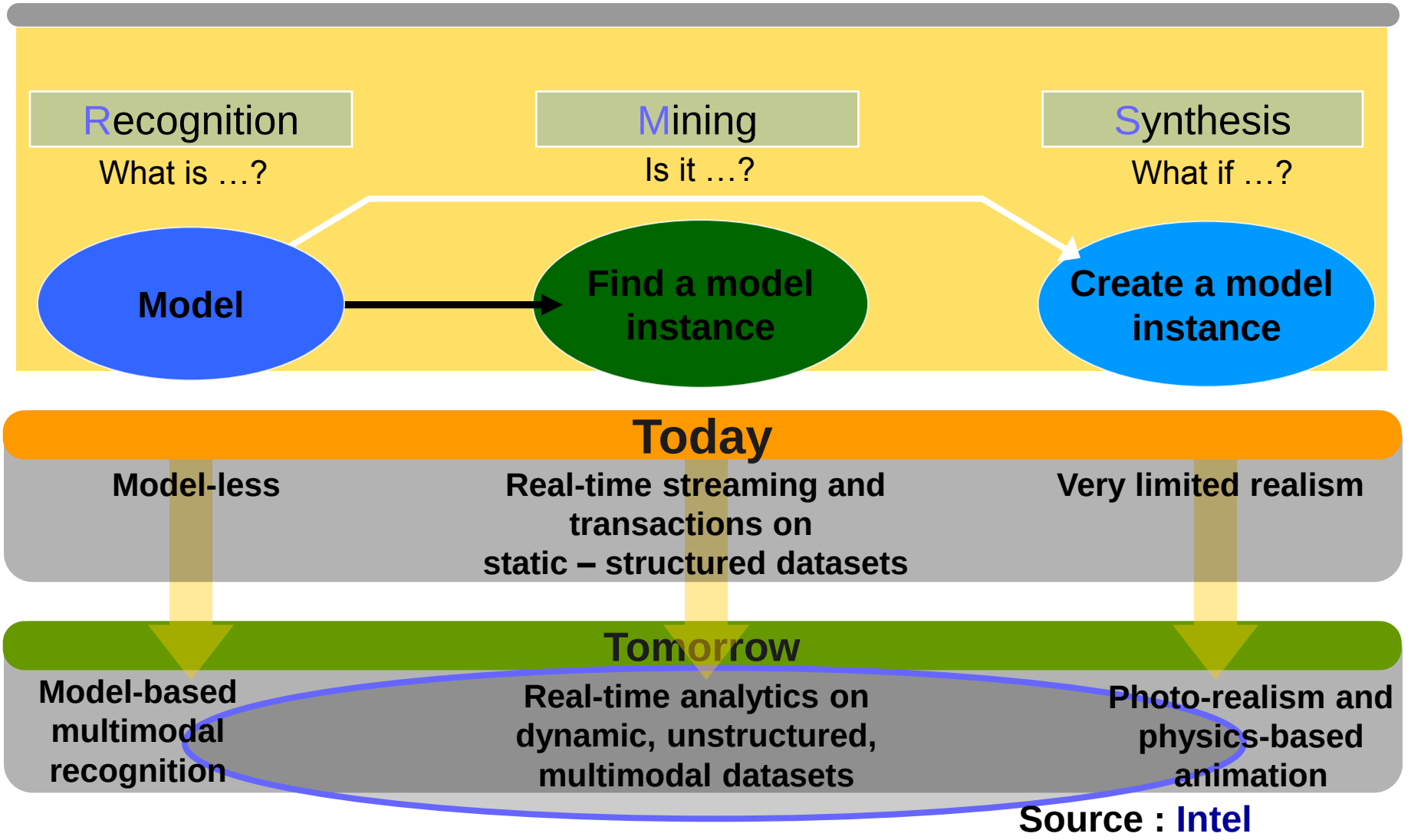
"Where is it?"
Search for a similar instance



Demo: Personal image search

Source: Intel

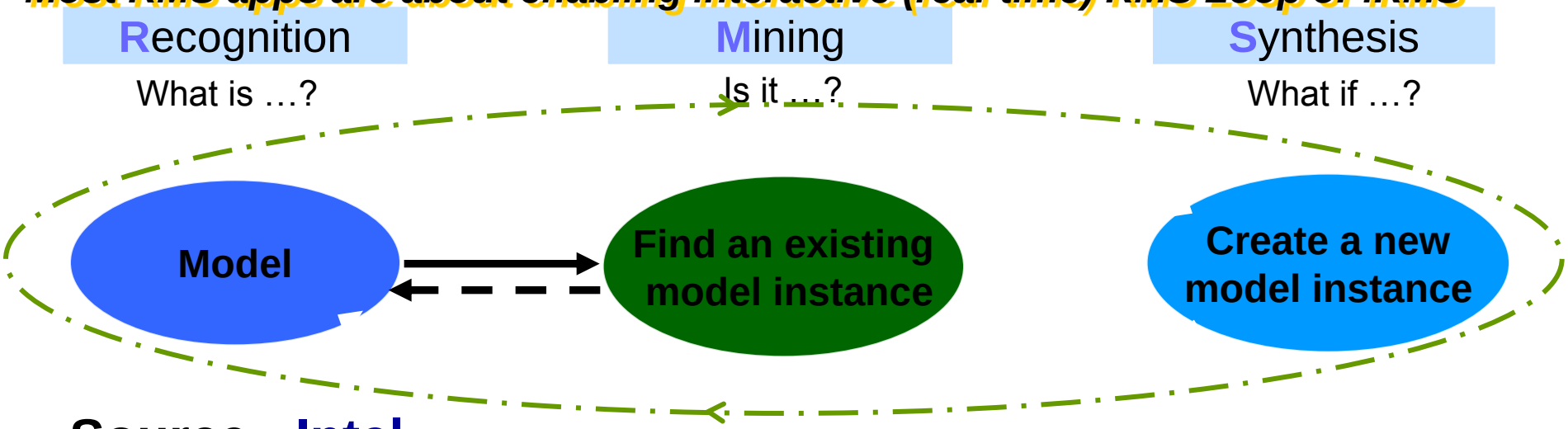
Emerging "Killer Apps of Tomorrow"
RMS: Recognition Mining & Synthesis



Source : Intel

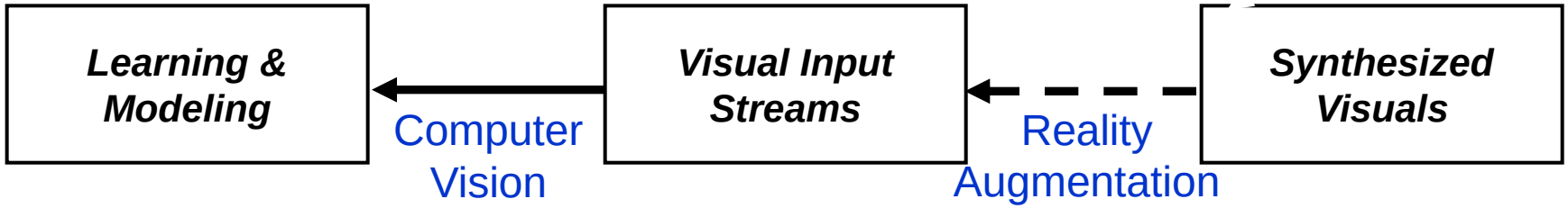
Emerging "Killer Apps of Tomorrow" Interactive Recognition Mining & Synthesis

Most RMS apps are about enabling interactive (real-time) RMS Loop or iRMS



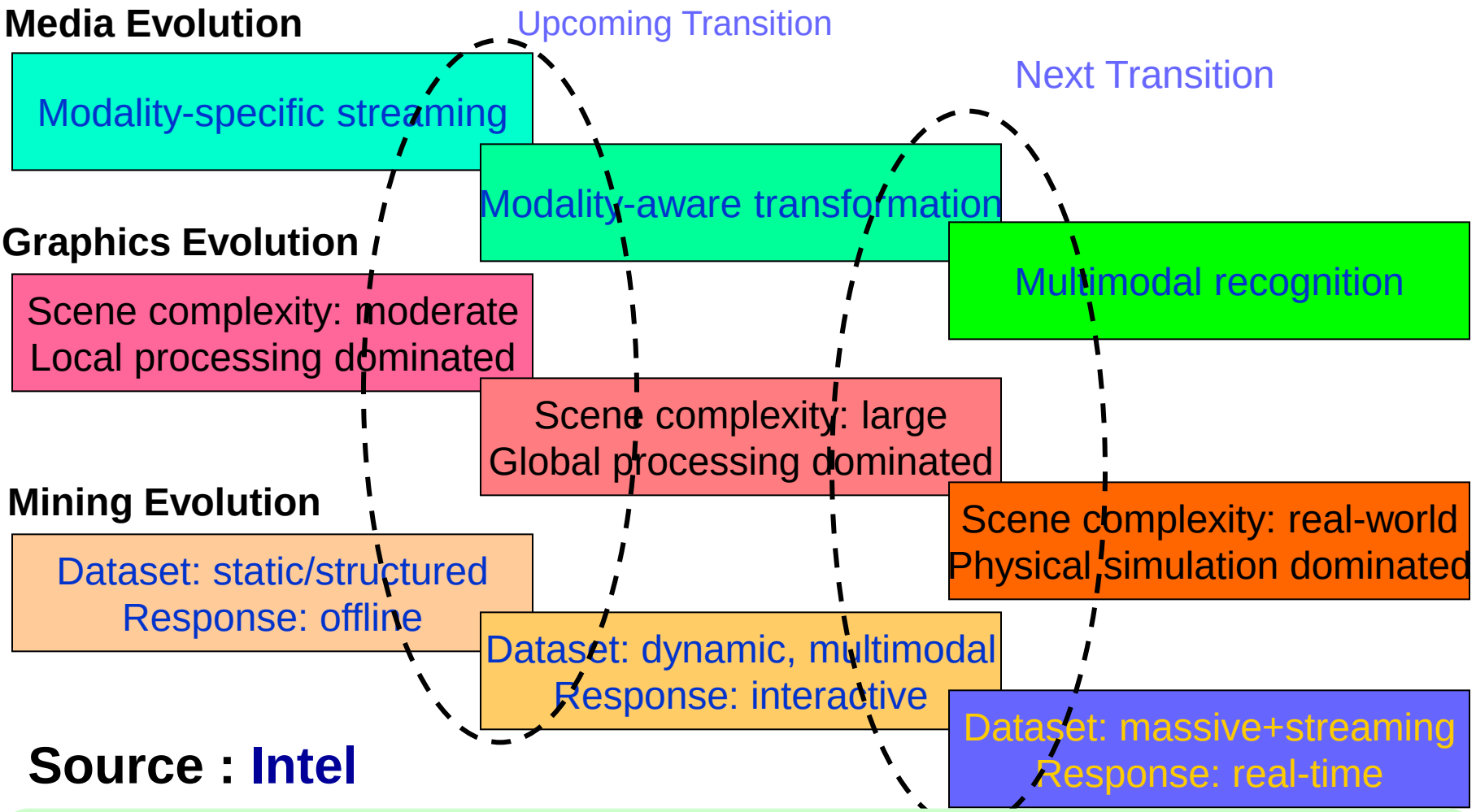
Source : Intel

Graphics Rendering + Physical Simulation



Workload convergence: multimodal recognition and synthesis over complex datasets

Emerging "Killer Apps of Tomorrow"
Interactive Recognition Mining & Synthesis



Source : Intel

Workload convergence: multimodal recognition and synthesis over complex datasets

Emerging "Killer Apps of Tomorrow" Parallel Algorithms Design

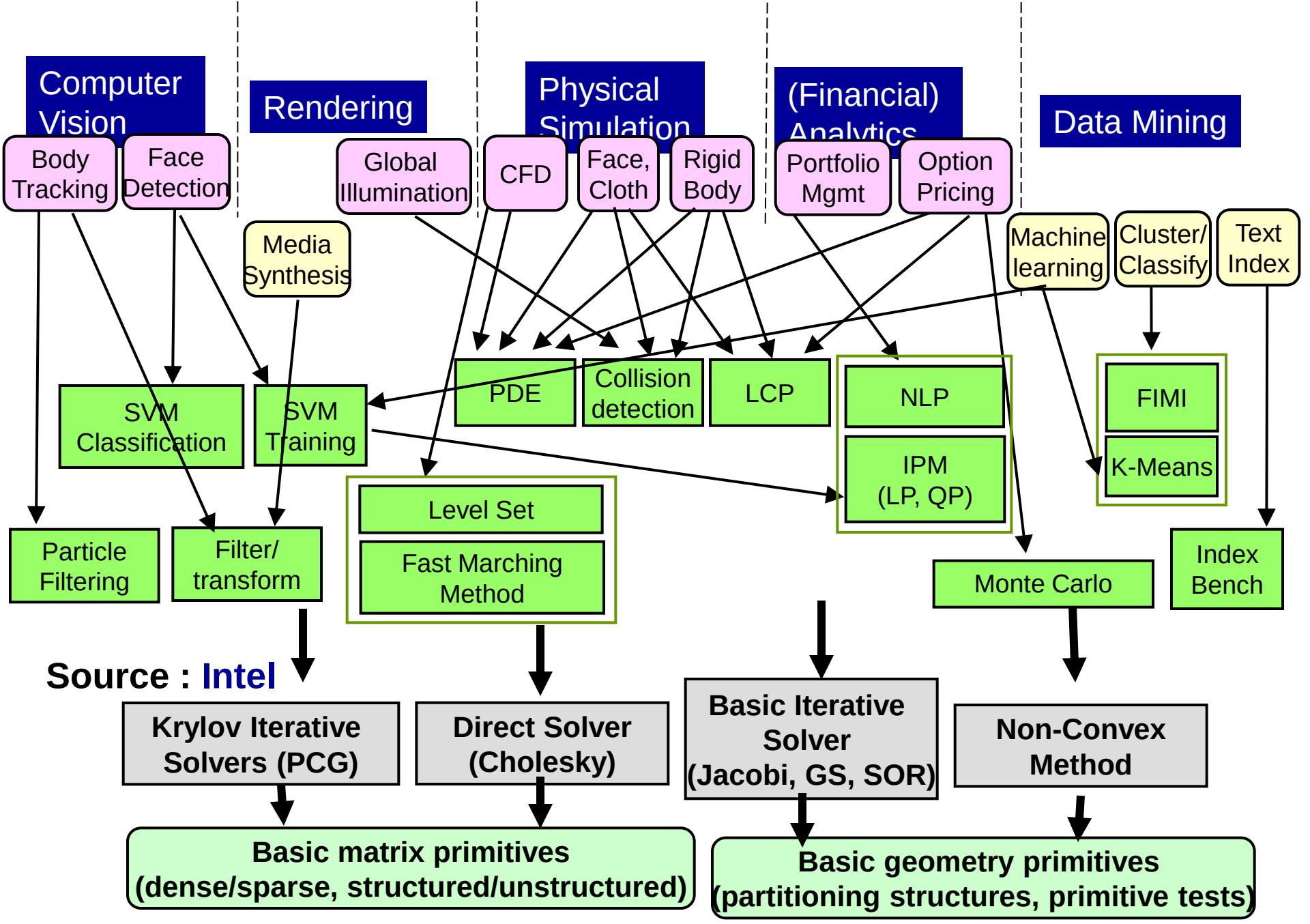
(Contd…)

- Video : Body Tracking and Ray-Tracing
- Search for Video based imaging & Creation of Videos with animation
- Games / Interactive 3D simulation
- Real-time realistic 3D Visualization of body systems
- Wall Street Financial Analysis; Business: Data Mining – Security: Real Time Video surveillance Astrophysics,
- BioInformatics – Pattern Search Algorithms

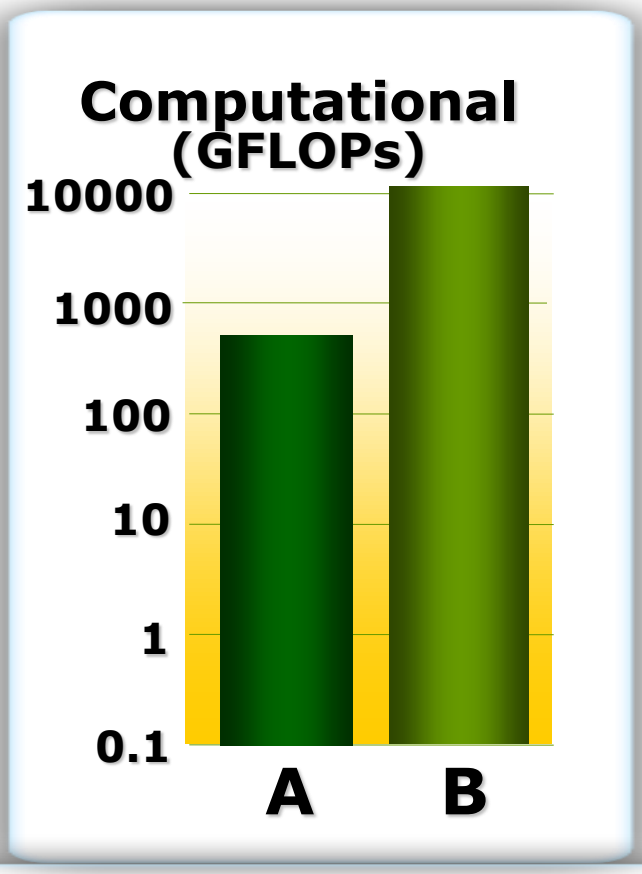
Emerging "Killer Apps of Tomorrow" Parallel Algorithms Design

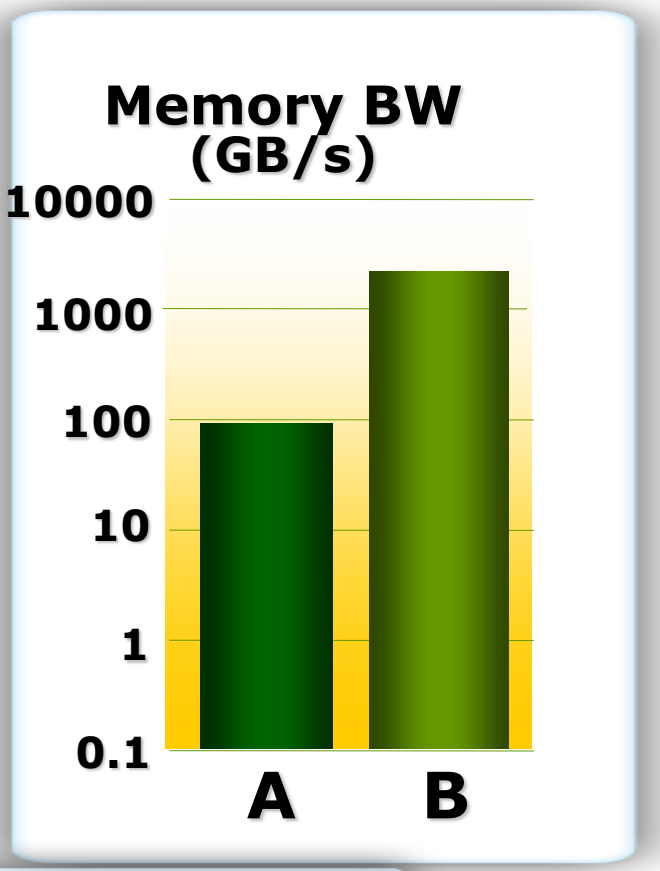
(Contd…)

- Speech Recognition & Speech Analysis – Modelling and Identifying using multi-modal data
- Large Scale image searches Audi /Video Control Systems
- Ray Tracing Algorithms – Video & Physical Simulation
- Overlapping computations with interactions.
- Data Mining Algorithms


Computer Vision
Rendering
Physical Simulation
(Financial) Analytics
Data Mining
Body Tracking
Face Detection
Global Illumination
CFD
Face, Cloth
Rigid Body
Portfolio Mgmt
Option Pricing
Machine learning
Cluster/ Classify
Text Index
Media Synthesis
SVM Classification
SVM Training
PDE
Collision detection
LCP
NLP
IPM (LP, QP)
FIMI
K-Means
Particle Filtering
Filter/ transform
Level Set
Fast Marching Method
Monte Carlo
Index Bench
Source : Intel
Krylov Iterative Solvers (PCG)
Direct Solver (Cholesky)
Basic Iterative Solver (Jacobi, GS, SOR)
Non-Convex Method
Basic matrix primitives (dense/sparse, structured/unstructured)
Basic geometry primitives (partitioning structures, primitive tests)

Emerging “Killer Apps of Tomorrow” Simulation



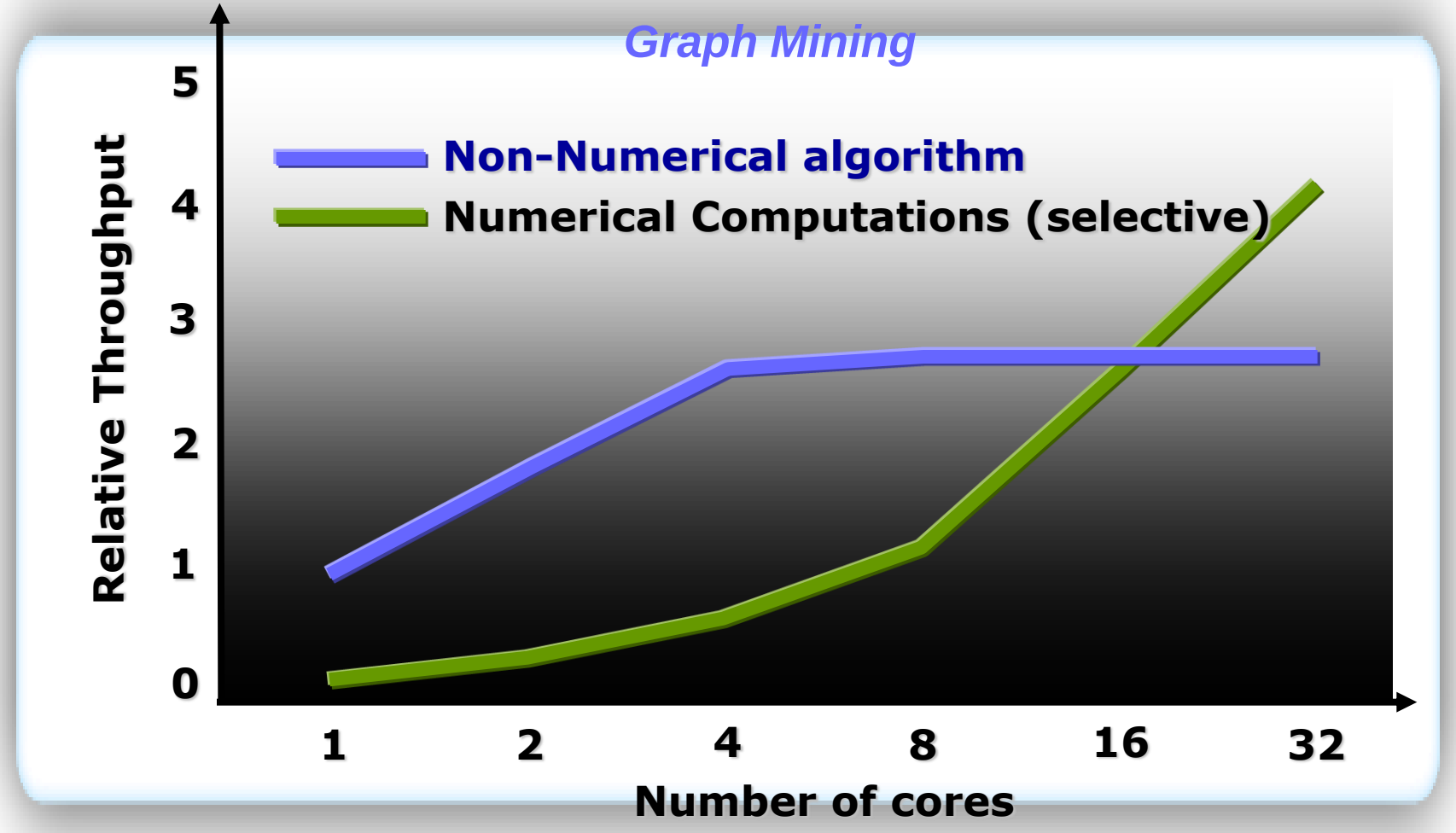


A: CFD, 75x50x50, 10 fps

B: CFD, 150x100x100, 30 fps

Source : Intel

Performance Analysis



Emerging "Killer Apps of Tomorrow":Summary

- RMS applications require significant increase in compute density
  - often non-linear extensions of existing usages
- RMS apps have a *converged common core of computing*
  - Workload convergence implies platform convergence
- Programmer productivity can be enhanced by platform
  - significant opportunity for silicon differentiation
- RMS apps will drive most future technology vectors
  - Programming to processor to memory technology
- Existing benchmarks are a poor proxy for these apps
  - Current popular suites must *evolve or go extinct!*

Emerging "Killer Apps of Tomorrow":References

- More web based info:
  - Justin Rattner, CTO, Intel, blog on "Cool Codes" http://blogs.zdnet.com/OverTheHorizon/
  - RMS Intro: http://www.intel.com/technology/magazine/computing/recognition-mining-synthesis-0205.htm
  - Platform 2015:
    - http://www.intel.com/technology/architecture/platform2015/

Source : Reference : [6]
http://www.intel.com

Summary

- An overview of Application Requirements for Multi-Core Systems
- An Overview of Threading

References

1. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Programming, Boston, MA : Addison-Wesley

2. Butenhof, David R **(1997),** Programming with POSIX Threads , Boston, MA : Addison Wesley Professional

3. Culler, David E., Jaswinder Pal Singh **(1999),** Parallel Computer Architecture - A Hardware/Software Approach , San Francsico, CA : Morgan Kaufmann

4. Grama Ananth, Anshul Gupts, George Karypis and Vipin Kumar **(2003),** Introduction to Parallel computing, Boston, MA : Addison-Wesley

5. Intel Corporation, **(2003),** Intel Hyper-Threading Technology, Technical User's Guide, Santa Clara CA : Intel Corporation Available at : http://www.intel.com

6. Shameem Akhter, Jason Roberts **(April 2006),** Multi-Core Programming - Increasing Performance through Software Multi-threading , Intel PRESS, Intel Corporation,

7. Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell **(1996),** Pthread Programming O'Reilly and Associates, Newton, MA 02164,

8. James Reinders, Intel Threading Building Blocks – (**2007**) , O'REILLY series

9. Laurence T Yang & Minyi Guo (Editors), (**2006**) *High Performance Computing - Paradigm and Infrastructure* Wiley Series on Parallel and Distributed computing, Albert Y. Zomaya, Series Editor

10. Intel Threading Methodology ; Principles and Practices Version 2.0 copy right **(March 2003),** Intel Corporation

References

11. William Gropp, Ewing Lusk, Rajeev Thakur **(1999),** Using MPI-2, Advanced Features of the Message-Passing Interface, The MIT Press..

12. Pacheco S. Peter, **(1992),** Parallel Programming with MPI, , University of Sanfrancisco, Morgan Kaufman Publishers, Inc., Sanfrancisco, California

13. Kai Hwang, Zhiwei Xu, (**1998**), Scalable Parallel Computing (Technology Architecture Programming), McGraw Hill New York.

14. Michael J. Quinn (**2004**), Parallel Programming in C with MPI and OpenMP McGraw-Hill International Editions, Computer Science Series, McGraw-Hill, Inc. Newyork

15. Andrews, Grogory R. **(2000),** Foundations of Multithreaded, Parallel, and Distributed Progrmaming, Boston, MA : Addison-Wesley

16. SunSoft. Solaris multithreaded programming guide. SunSoft Press, Mountainview, CA, **(1996),** Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill,

17. Chandra, Rohit, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon, **(2001)**,Parallel Programming in OpenMP San Fracncisco Moraan Kaufmann

18. S.Kieriman, D.Shah, and B.Smaalders **(1995),** Programming with Threads, SunSoft Press, Mountainview, CA. 1995

19. Mattson Tim, **(2002),** Nuts and Bolts of multi-threaded Programming Santa Clara, CA : Intel Corporation, Available at : http://www.intel.com

20. I. Foster **(1995,** Designing and Building Parallel Programs ; Concepts and tools for Parallel Software Engineering, Addison-Wesley (1995)

21. J.Dongarra, I.S. Duff, D. Sorensen, and H.V.Vorst **(1999),** Numerical Linear Algebra for High Performance Computers (Software, Environments, Tools) SIAM, 1999

References

22. OpenMP C and C++ Application Program Interface, Version 1.0". **(1998),** OpenMP Architecture Review Board. October 1998

23. D. A. Lewine. *Posix Programmer's Guide:* **(1991),** Writing Portable Unix Programs with the Posix. 1 Standard. O'Reilly & Associates, 1991

24. Emery D. Berger, Kathryn S McKinley, Robert D Blumofe, Paul R.Wilson, *Hoard : A Scalable Memory Allocator for Multi-threaded Applications* ; The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX). Cambridge, MA, November (**2000)**. Web site URL : http://www.hoard.org/

25. Marc Snir, Steve Otto, Steyen Huss-Lederman, David Walker and Jack Dongarra, (**1998**) *MPI-The Complete Reference: Volume 1, The MPI Core, second edition* [MCMPI-07].

26. William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Marc Snir (**1998**) *MPI-The Complete Reference: Volume 2, The MPI-2 Extensions*

27. A. Zomaya, editor. Parallel and Distributed Computing Handbook. McGraw-Hill, **(1996)**

28. OpenMP C and C++ Application Program Interface, Version 2.5 (**May 2005**)", From the OpenMP web site, URL **: http://www.openmp.org/**

29. Stokes, Jon 2002 Introduction to Multithreading, Super-threading and Hyper threading *Ars Technica*, October **(2002)**

30. Andrews Gregory R. 2000, Foundations of Multi-threaded, Parallel and Distributed Programming, Boston MA : Addison – Wesley (**2000)**

31. Deborah T. Marr , Frank Binns, David L. Hill, Glenn Hinton, David A Koufaty, J . Alan Miller, Michael Upton, "Hyperthreading, Technology Architecture and Microarchitecture", Intel (**2000-01)**

- **Thank You**
- *Any questions ?*