Performance Productivity Challenges and Researches

Victor Lee

Parallel Computing Lab, Intel

Acknowledgement: The contributions from Youfeng Wu from PSL, Intel and other members of PCL Intel

Agenda

- Performance Productivity Gap
- What Created the Gap
- Past and Current Research
- Summary

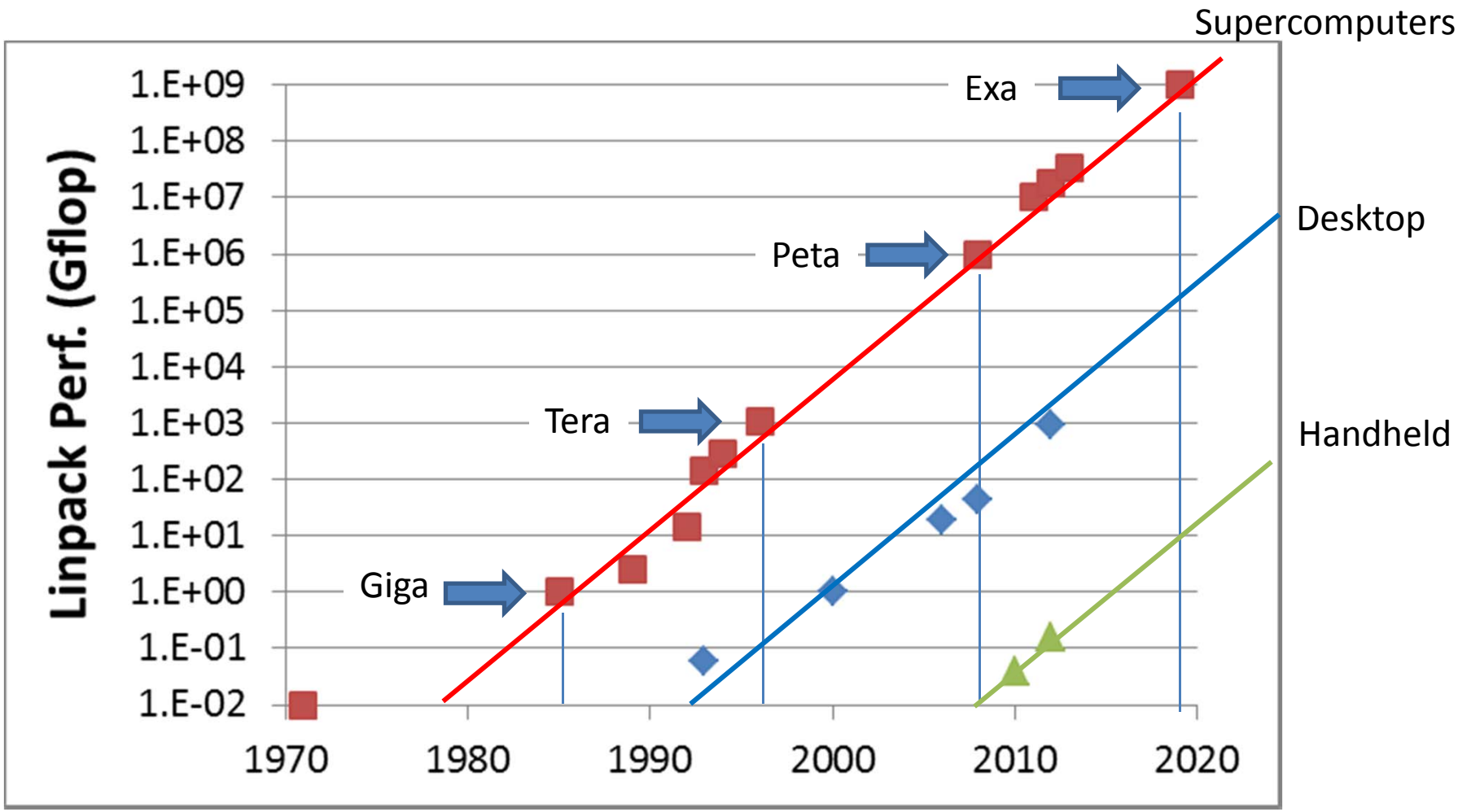
Copyright victor.w.lee@intel.com

Agenda

- Performance Productivity Gap
- What Created the Gap
- Past and Current Research
- Summary

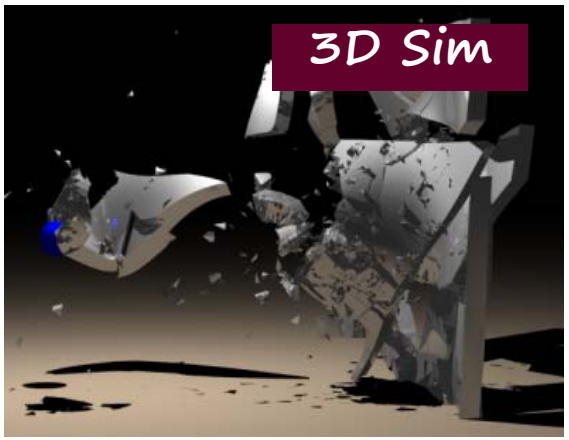
Copyright victor.w.lee@intel.com

Compute Performance Roadmap



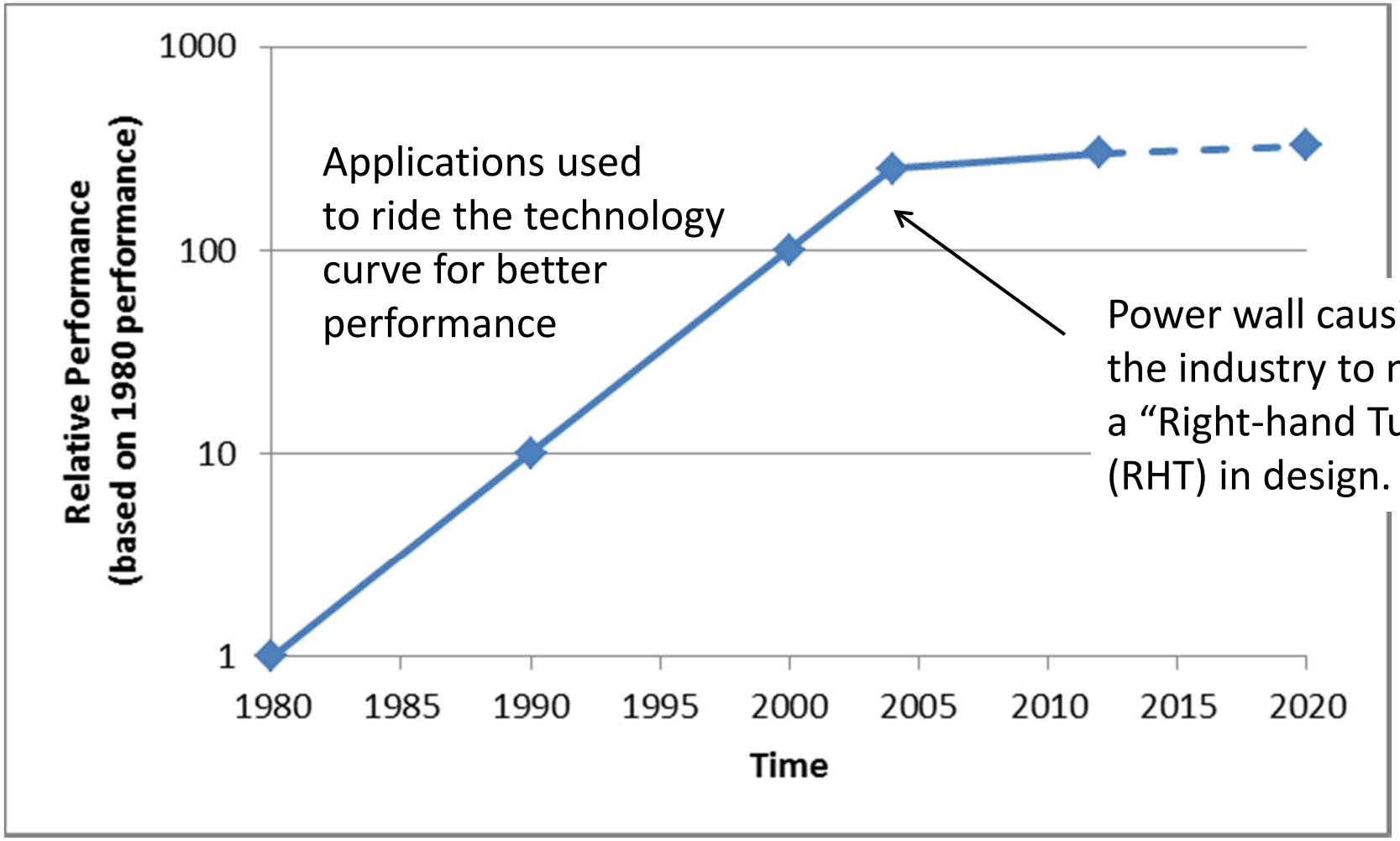
Copyright victor.w.lee@intel.com

Endless Opportunities



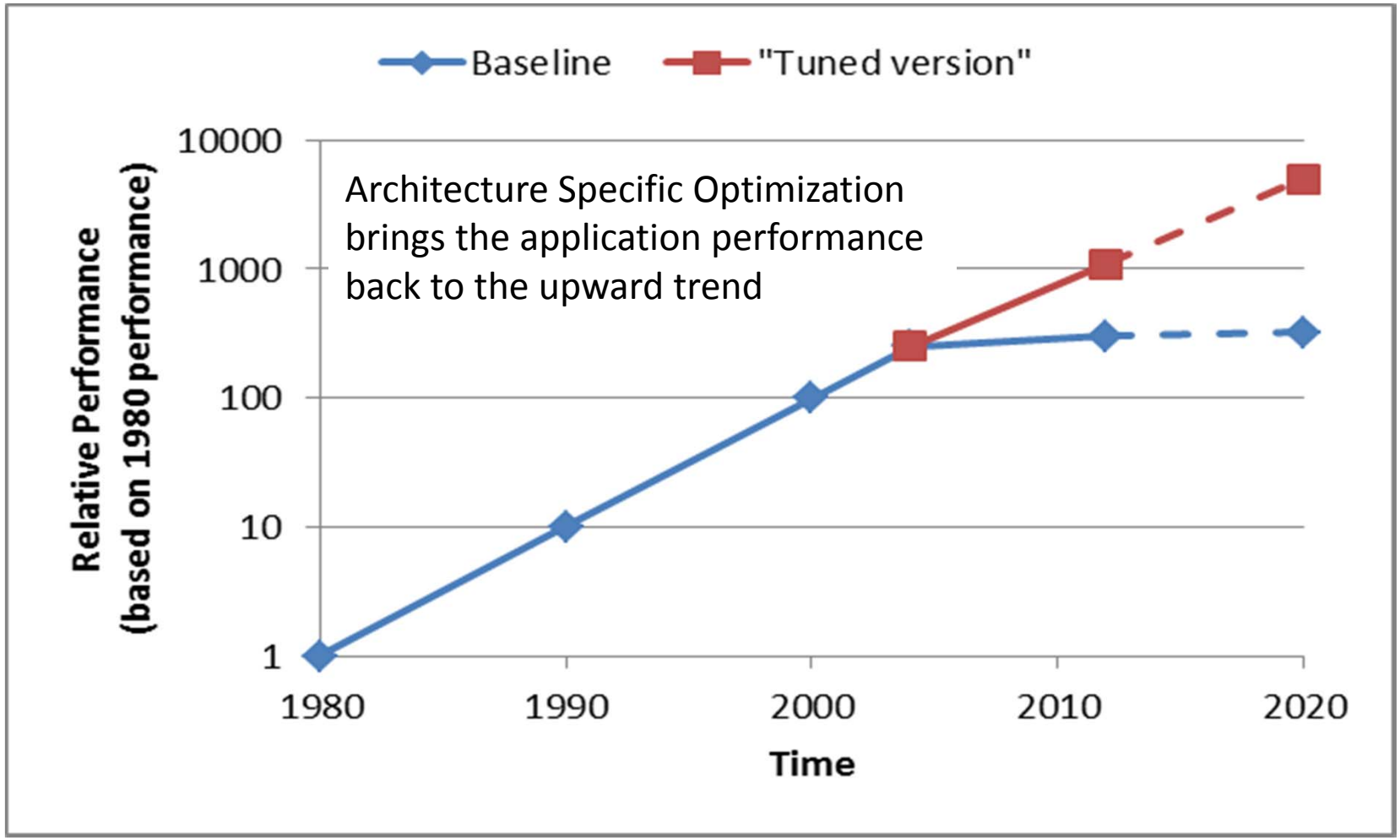
Copyright victor.w.lee@intel.com

Application Performance Implications



Copyright victor.w.lee@intel.com

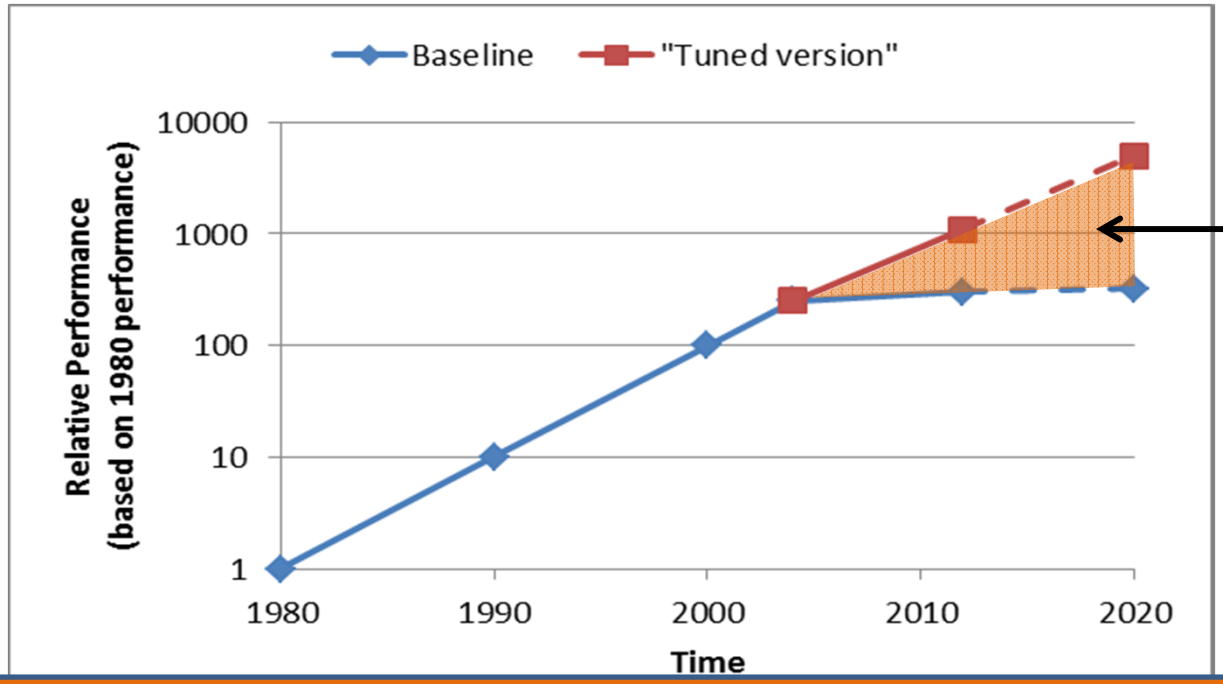
Architecture Specific Optimization



Copyright victor.w.lee@intel.com

Performance Productivity Gap

- Definition: performance difference between existing software and the optimized software



"Performance-Productivity gap"

"Performance Productivity Gap" = Opportunity Lost

Copyright victor.w.lee@intel.com

Mini-Summary 1

- Moore's law is alive and well. Future processors to have many cores and great performance potential
- Current SW experience "Performance Productivity Gap" and can lead to competitive disadvantages

Copyright victor.w.lee@intel.com

Agenda

- Performance Productivity Gap
- What Created the Gap
- Past and Current Research
- Summary

Copyright victor.w.lee@intel.com

Sources of Performance Productivity Gap

- The obvious:
  - Many cores
  - The memory

- The not so obvious:
  - Energy efficiency challenge
  - Heterogeneity
  - Core variations
  - Failures

Copyright victor.w.lee@intel.com

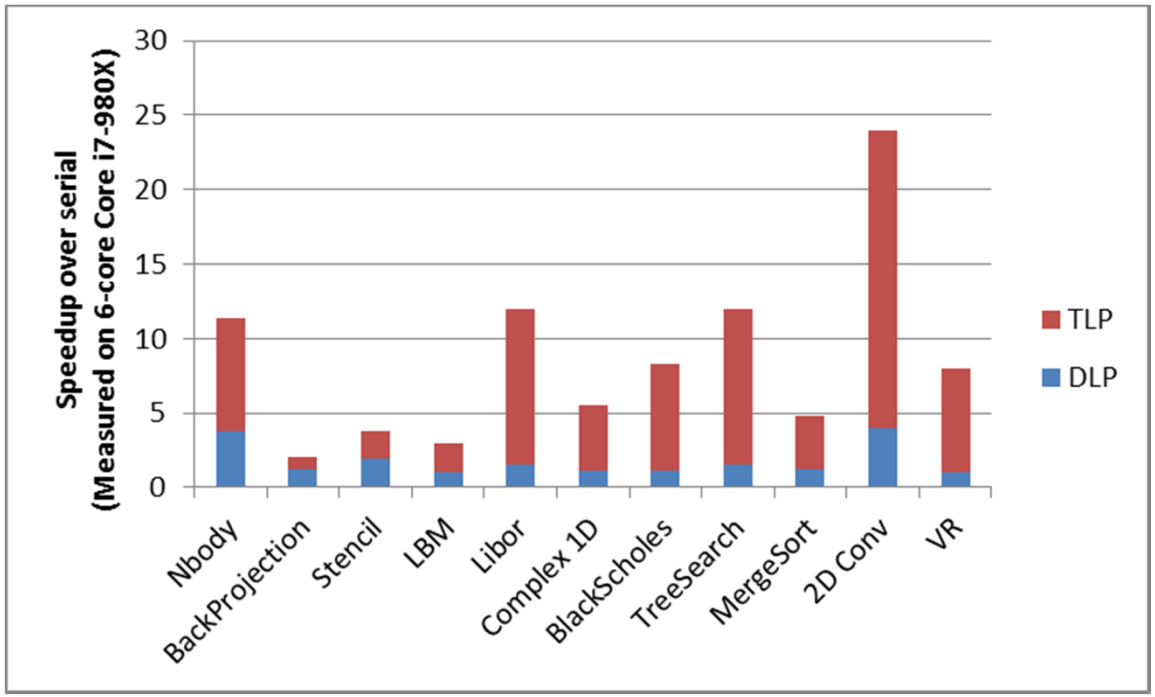
Problem w/ Parallel Computing

- Parallel systems
  - In the past, are for the HPC programmers
  - Result of the industry “RHT”, parallel systems are for everyone now
- Why parallel computing is hard
  - People tends to think sequentially
  - Very few are taught to write parallel programs
  - Simply extending serial programs will not get good performance and will be hard to debug

Copyright victor.w.lee@intel.com

Parallel Opportunity / Challenges

Exploring multi-core and SIMD can result in significant speedup



* Data drawn from Satish et. al. "Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications?", In proc. of ISCA 2012

Thread Level Parallelism (TLP) Challenges:

- Think concurrency
- Data/task decomposition
- Synchronization

Data Level Parallelism (DLP) Challenges:

- Data alignment
- Control flow divergence

Copyright victor.w.lee@intel.com

Problem in Feeding

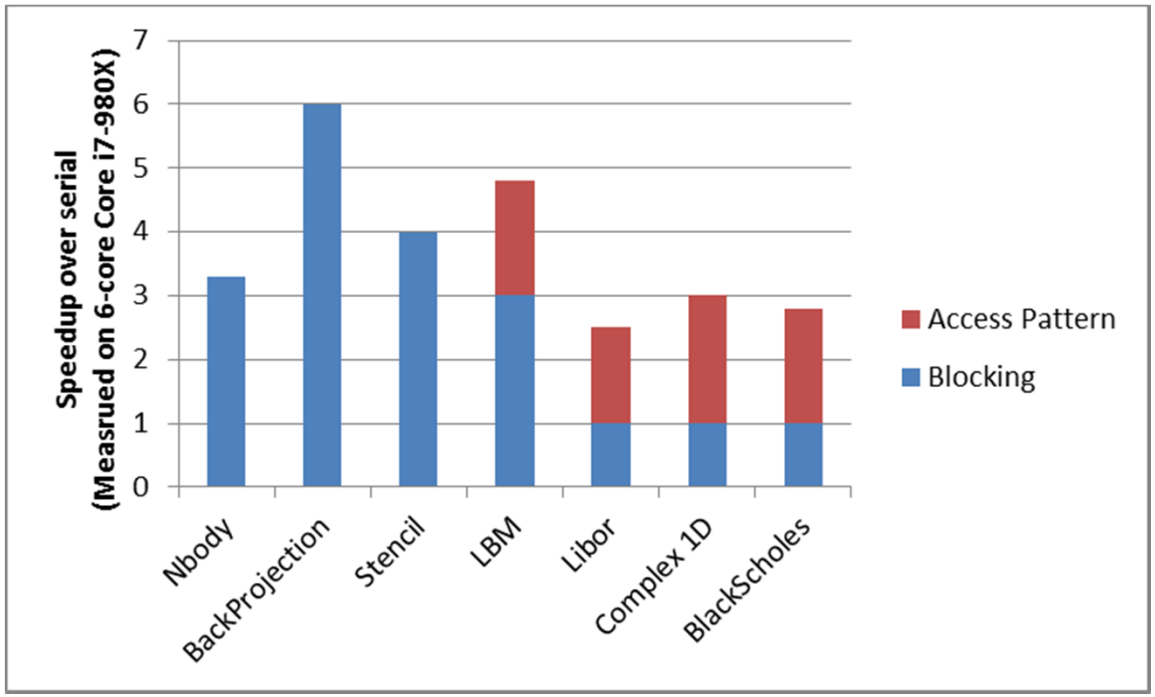
- Feeding one core is hard enough
  - Memory performance lags processor performance
    - Survey shows: Processor improves 50% a year, memory BW improves ~20%, and memory latency improves ~5% a year

- Feeding many cores is much harder
  - Not enough BW to go around
  - Conflicts and contention

Copyright victor.w.lee@intel.com

Data Access Opportunity / Challenges

Optimizing data access patterns and making use of cache / local memory can mitigate data access problem



* Data drawn from Satish et. al. "Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications?", In proc. of ISCA 2012

Challenges:

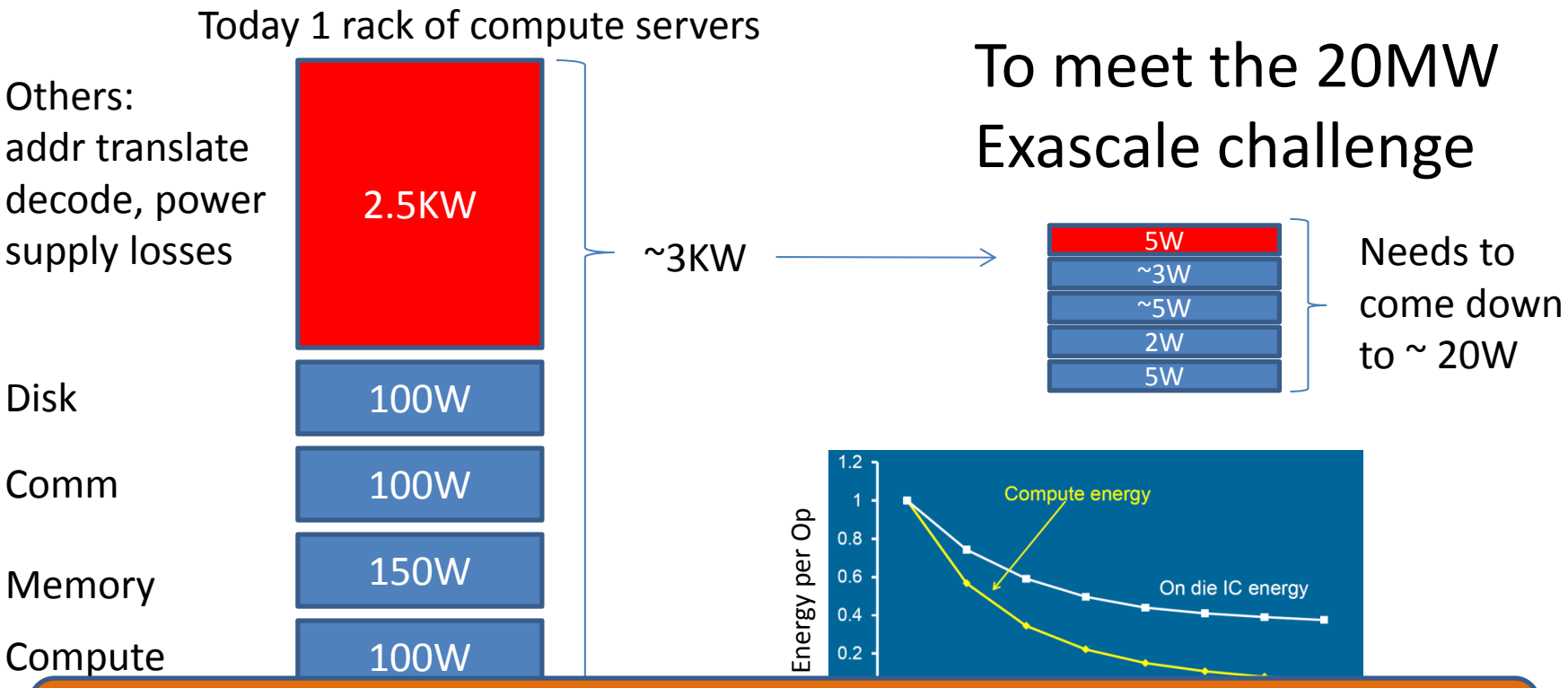
- No standardize memory hierarchy
- Intuitive data structure is not necessary optimal for modern memory subsystem

Copyright victor.w.lee@intel.com

THE NOT SO OBVIOUS CAUSES

Energy Efficiency Challenge

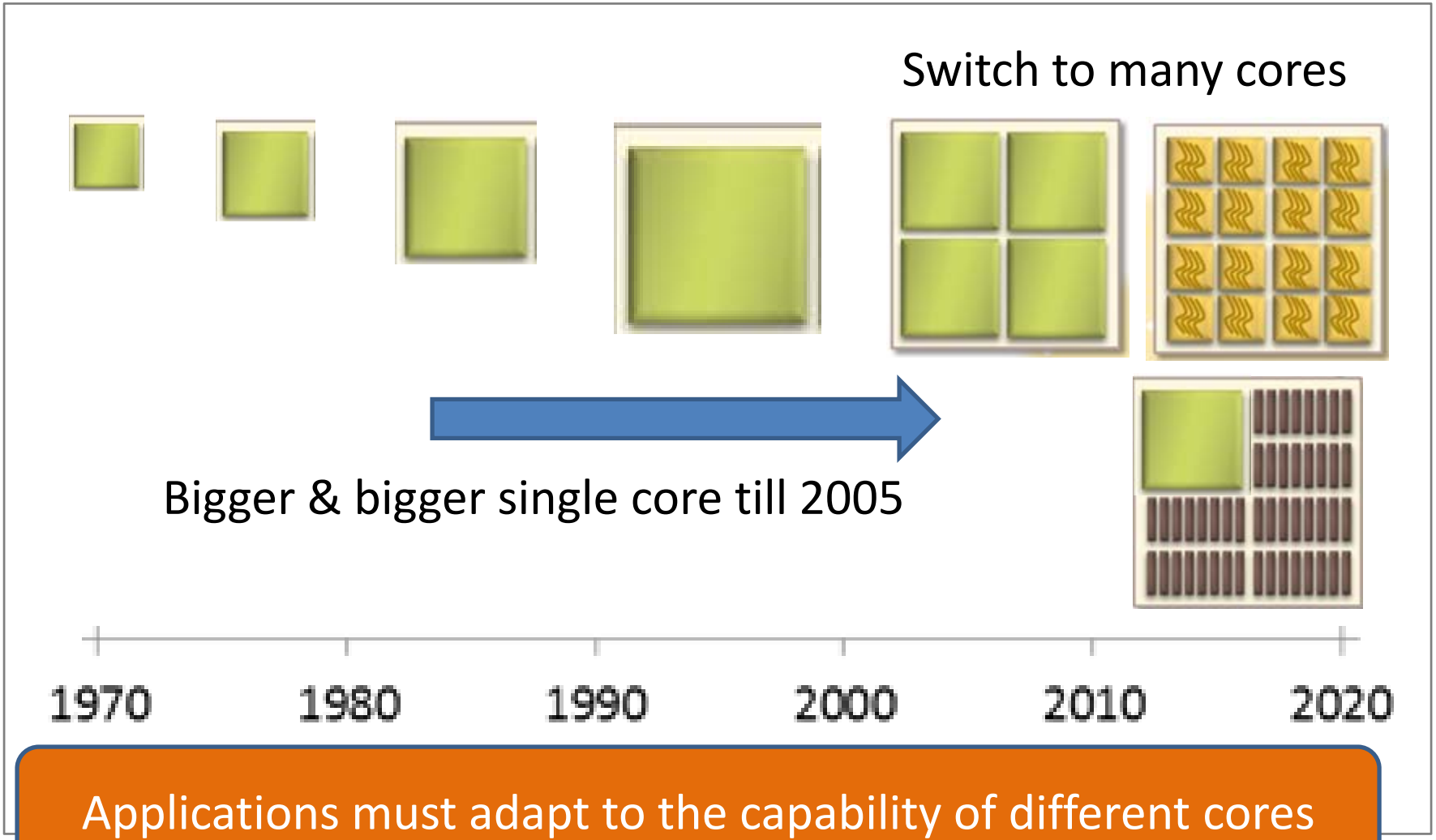
- Today 50PF computer at 30MW power



More efficient application can help energy issue

Copyright victor.w.lee@intel.com

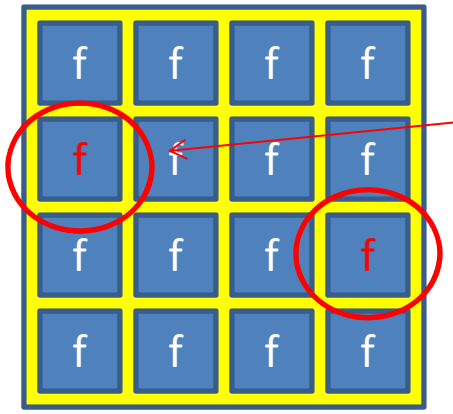
Heterogeneity is Here



Applications must adapt to the capability of different cores

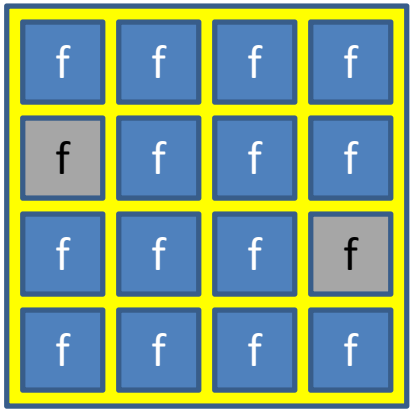
Copyright victor.w.lee@intel.com

Manufacturing Variations

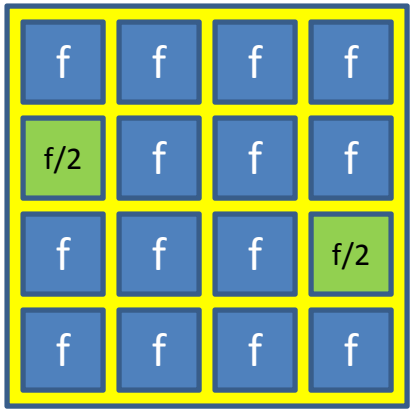


Some cores will inherently runs slower

Soln1: Turn them off



Soln2: Turn them @ diff freq

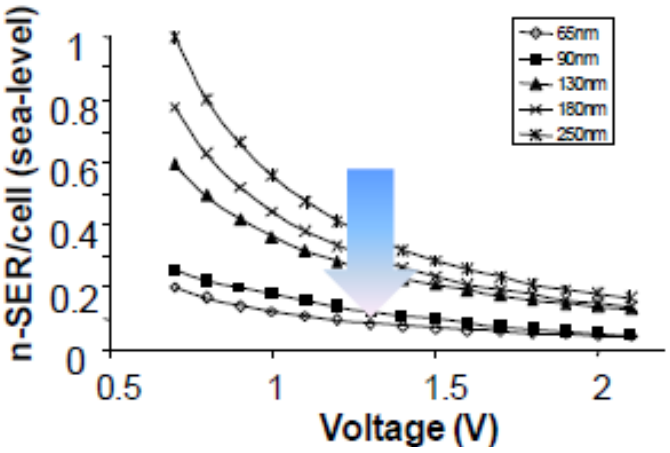


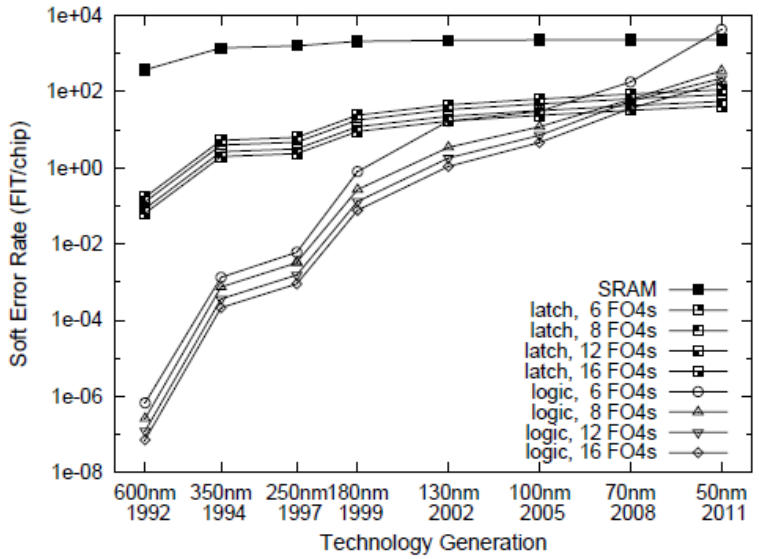
Again application must adapt to core variation

Copyright victor.w.lee@intel.com

Tolerating Faults

- Smaller feature sizes reduce soft error rate
- Increases in # of components per chip cause overall error rate to increase





Premkishore Shivakumar, et al. *Modeling the Impact of Device and Pipeline Scaling on the Soft Error Rate of Processor Elements*. Computer Science Department, University of Texas at Austin, 2002.

Copyright victor.w.lee@intel.com

Mini-Summary 2

- An array of technology challenges cause the performance productivity gap
- Significant performance opportunity but lots of challenges

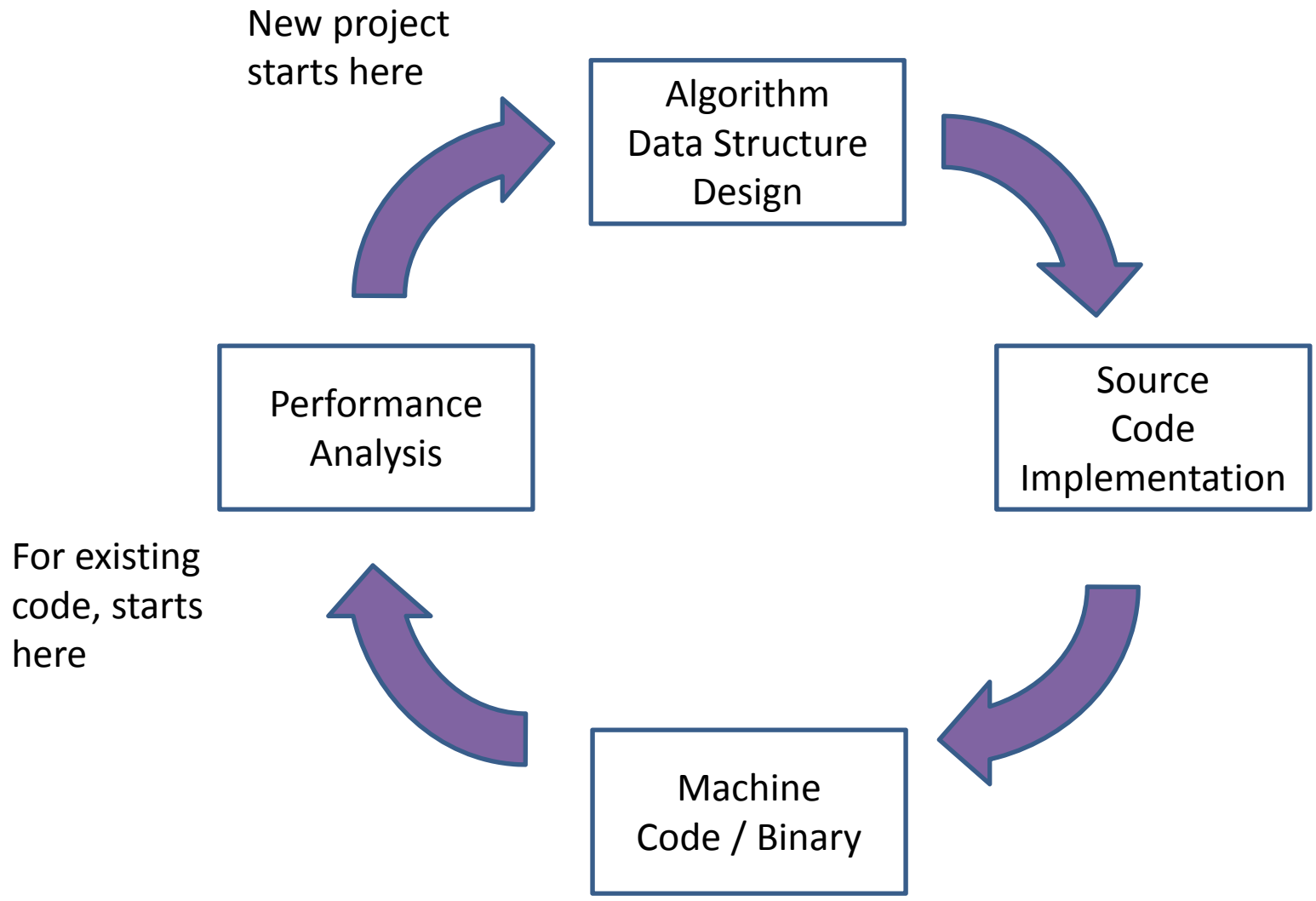
Copyright victor.w.lee@intel.com

Agenda

- Performance Productivity Gap
- What Created the Gap
- Past and Current Researches
- Summary

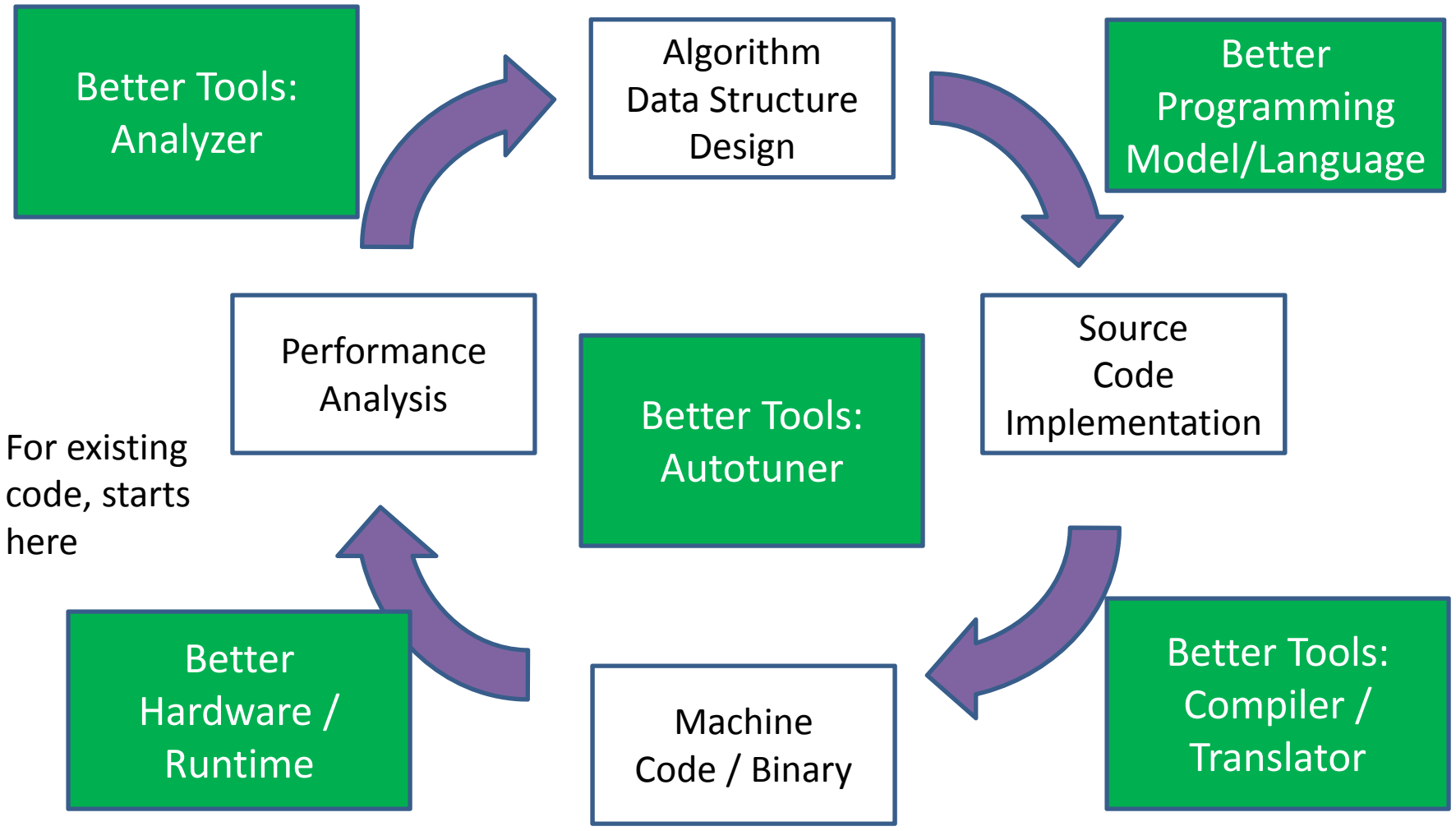
Copyright victor.w.lee@intel.com

Program Development Workflow



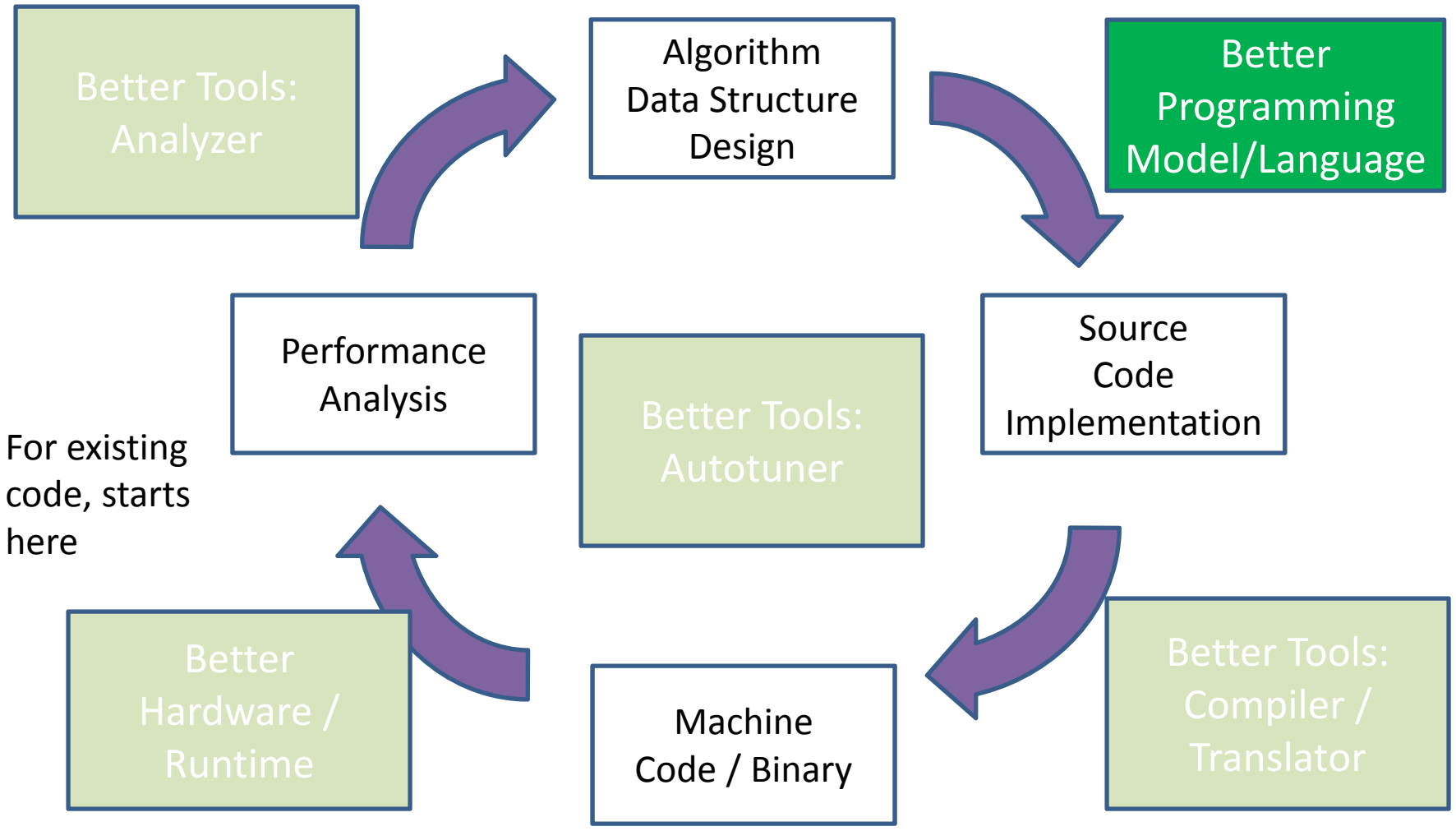
Copyright victor.w.lee@intel.com

Opportunities to Bridge the Performance Productivity Gap



Copyright victor.w.lee@intel.com

Programming Model Language & Perf-Productivity Gap



Copyright victor.w.lee@intel.com

Desired Properties

- Allow programmers to:
  - Express concurrency (at different levels)
  - Manage data locality
  - Provide determinism to aid debug
- Other goodies:
  - Portability across architecture
  - Easy code reuse
  - Distributed development

Copyright victor.w.lee@intel.com

Approaches

- Library extensions of existing sequential languages
  - E.g., SHMEM, MPI, PVM
- Directives based
  - OpenMP, OpenACC
- New parallel languages
  - Charm++, CILK, Co-Array Fortran, Titanium, UPC, X10, Fortress, Chapel, CUDA, OpenCL, Parallel JavaScript

Copyright victor.w.lee@intel.com

Chapel, Fortress and X10

	Chapel	Fortress	X10
Parallel model	Concurrent Tasks	Parallel Tasks	Concurrent Activities
Array Data Types / Pointers	Yes	Yes	Yes
Data management	PGAS w/ Locale	PGAS on arrays	PGAS w/ Place

```
config var n = 5,                       // size of n x n grid
           epsilon = 0.00001;           // convergence tolerance

def main() {
  const ProblemSpace = [1..n, 1..n],    // domain for grid points
        BigDomain = [0..n+1, 0..n+1];   // domain including boundary points

  var X, XNew: [BigDomain] real = 0.0;  // declare arrays:
                                        //   X stores approximate solution
                                        //   XNew stores the next solution

  X[n+1, 1..n] = 1.0;                   // Set south boundary values to 1.0

  var iteration = 0,                    // iteration counter
      delta: real;                      // measure of convergence

  const north = (-1,0), south = (1,0), east = (0,1), west = (0,-1);

  do {
    // compute next approximation using Jacobi method and store in XNew
    forall ij in ProblemSpace do
      XNew(ij) = (X(ij+north) + X(ij+south) + X(ij+east) + X(ij+west)) / 4.0;

    // compute difference between next and current approximations
    delta = max reduce abs(XNew[ProblemSpace] - X[ProblemSpace]);

    // update X with next approximation
    X[ProblemSpace] = XNew[ProblemSpace];

    // advance iteration counter
    iteration += 1;

  } while (delta > epsilon);

}
```

```
component fortress.executable

export Executable

run(args:String...):()=do

   needleLength = 20                          // declaration of
   numRows = 10                               // immutable variables
   tableHeight = needleLength numRows

   var hits : RR64 = 0.0                      // declaration of mutal
   var n : RR64 = 0.0                         // variables of type RI

   for i <- 1#3000 do                         // 3000 iterations
      delta_X = random(2.0) - 1
      delta_Y = random(2.0) - 1
      rsq = delta_X^2 + delta_Y^2

      if 0 < rsq < 1 then
         y1 = tableHeight random(1.0)
         y2 = y1 + needleLength (delta_Y / sqrt(rsq))
         (y_L, y_H) = (y1 MIN y2, y1 MAX y2)

         // increase 'hits' if needle hits line
         if ceiling(y_L/needleLength) = floor(y_H/needleLength) then
            atomic do hits += 1.0 end
         end
         atomic do n += 1.0 end
      end
   end

   probability = hits/n
   pi_est = 2.0/probability
   end
end
```

```
public class Jacobi extends x10Test {

    const int N = 5;                            // size of grid
    const double epsilon = 0.0001;              // convergence tolerance
    const double epsilon2 = 0.000000001;

    const region(:rank==2) RInner = [1:N, 1:N]; // region for grid points
    const region(:rank==2) R = [0:N+1, 0:N+1];  // region including boundary
                                                // points

    // distribution of grid
    const dist(:rank==2) D = (dist(:rank==2)) dist.factory.block(R);
    const dist(:rank==2) DInner = D | RInner;    // distribution for inner region of grid
    const dist(:rank==2) DBoundary = D - RInner; // boundary region of grid

    const int EXPECTED_ITERS = 97;
    const double EXPECTED_ERR = 0.0018673382039402497;

    final double[.] XNew = new double[D] (point p[i,j]){
        return DBoundary.contains(p) ? (N-1)/2 : N*(i-1)+(j-1);
    };

    public boolean run() {
        int iters = 0;
        double err;
        while (true) {
            final double[:distribution==this.DInner] X = new double[DInner] (point [i,j]){
                return (XNew[i+1,j]+XNew[i-1,j]+XNew[i,j+1]+XNew[i,j-1])/4.0;
            };
            if ((err = ((XNew | this.DInner)-X).abs().sum()) < epsilon) break;
            XNew.update(X);
            iters++;
        }
        System.out.println("Error = "+err);
        System.out.println("Iterations = "+iters);
        return Math.abs(err-EXPECTED_ERR) < epsilon2 && iters == EXPECTED_ITERS;
    }

    public static void main(String[] args) {
        new Jacobi().execute();
    }
}
```

Codes extracted from Michele Weiland, "Chapel, Fortress and X10: Novel languages for HPC", *The University of Edinburgh, Tech. Rep., October* (2007).

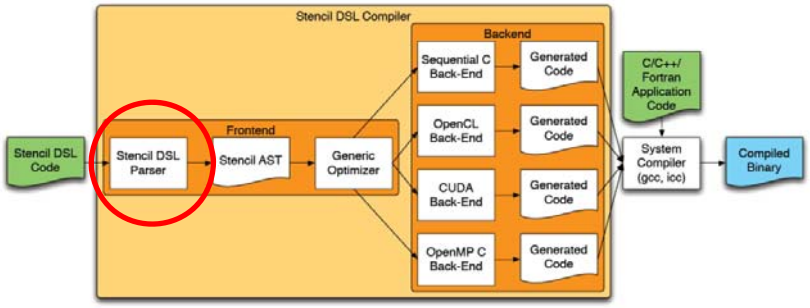
Copyright victor.w.lee@intel.com

Domain Specific Languages

DSLs trade off generality to better enable back-end tools performance and portability

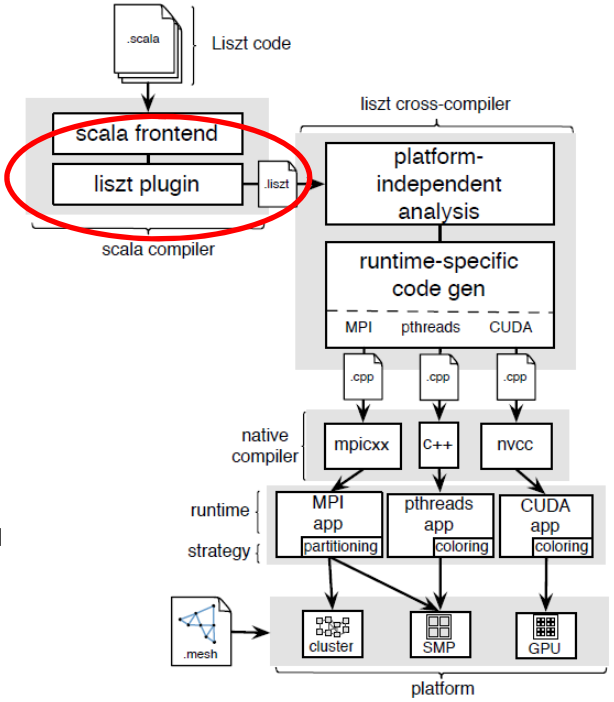
- Stencil DSL [Holewinski'12]

  Write stencil as point function and grid rather than loop-nest



Stencil Compiler Workflow

- Liszt [DeVito'11] : for mesh -based PDE solver

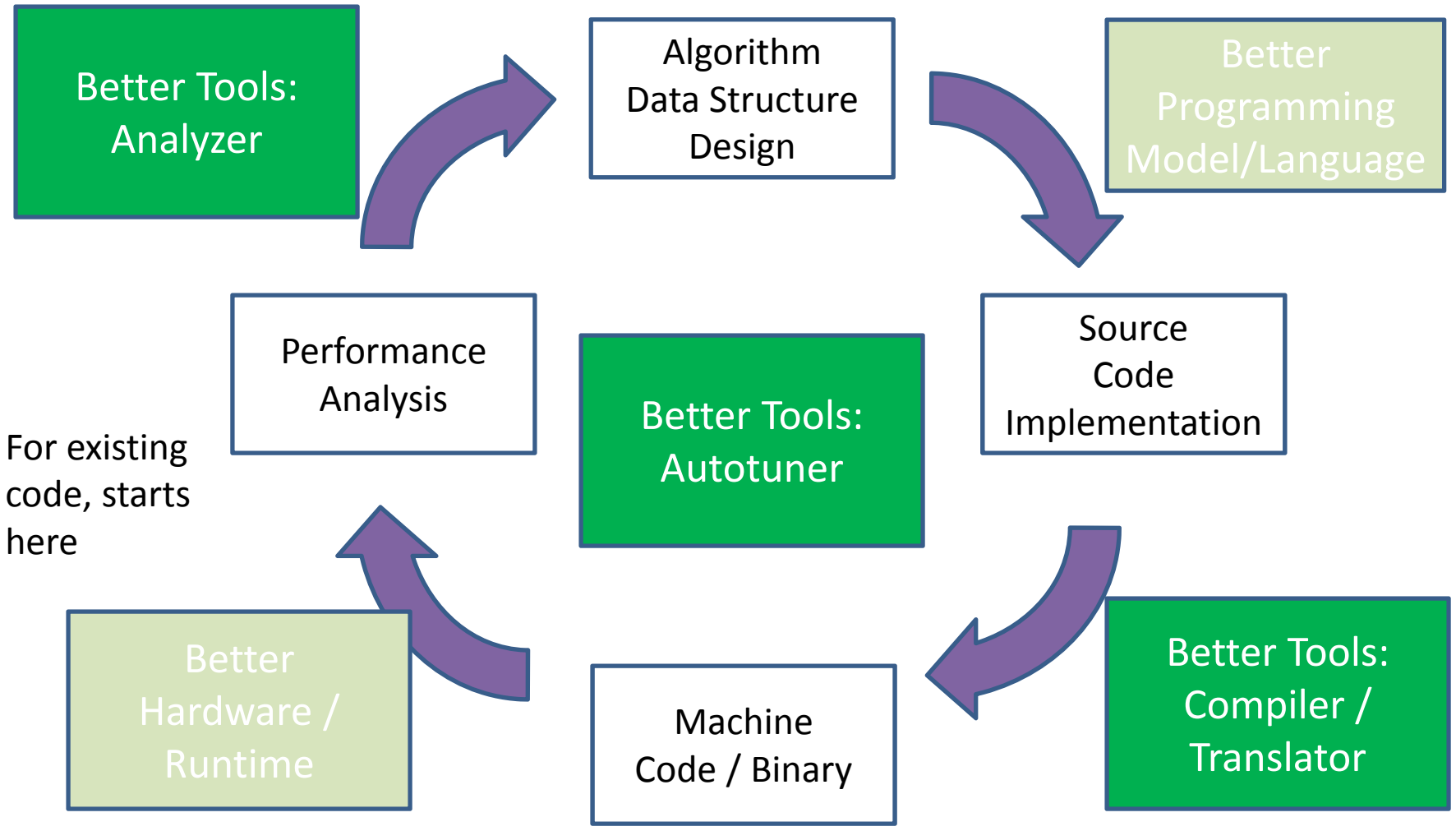


[Holewinski'12] Justin Holewinski, et al. "High-performance code generation for stencil computations on GPU architectures."*Proceedings of the 26th ACM international conference on Supercomputing*. ACM, 2012.

[DeVito'11] Zachary DeVito, et al. "Liszt: a domain specific language for building portable mesh-based PDE solvers." *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011.

Copyright victor.w.lee@intel.com

Tools & Perf-Productivity Gap



Copyright victor.w.lee@intel.com

Tools

Tools help manage the low level implementation details for parallelism & data management

- Compiler
- Analyzer
- Auto-tuner

Copyright victor.w.lee@intel.com

Compiler Optimizations

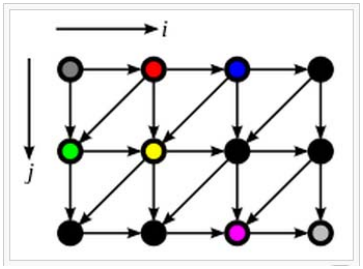
- Parallelization (and vectorization)
  - Focus on picking out independent tasks to form threads of execution through data/control dependency analysis
    - E.g., Independent loop iterations can be parallelized as different threads
  - More recent work:
    - Speculative parallelization – requires special HW support
    - Semi-automatic parallelization – requires programmers to help annotate the section that can be parallelized
  - Some reference: SUIF, Polaris, Helix, DSWP, Cray compiler, Intel C++ Compiler

Copyright victor.w.lee@intel.com

Compiler Optimizations

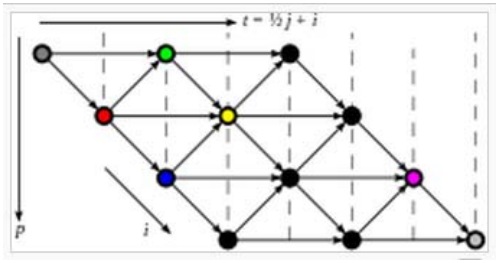
- Array padding and tiling
  - Padding to minimize access conflicts
  - Use static analysis to identify the optimal blocking factor to maximize reuse
  - Complications include: multiple level tiling, alignment, data placement, load balance

- Loop transformation:
  - Reshape the execution within a loop to improve program execution as well as data locality
  - State-of-the-art uses polyhedral models to handle a wider class of programs



j first, followed by i

A[i][j] <- A[i-1][j], A[i][j-1] and A[i+1][j-1]



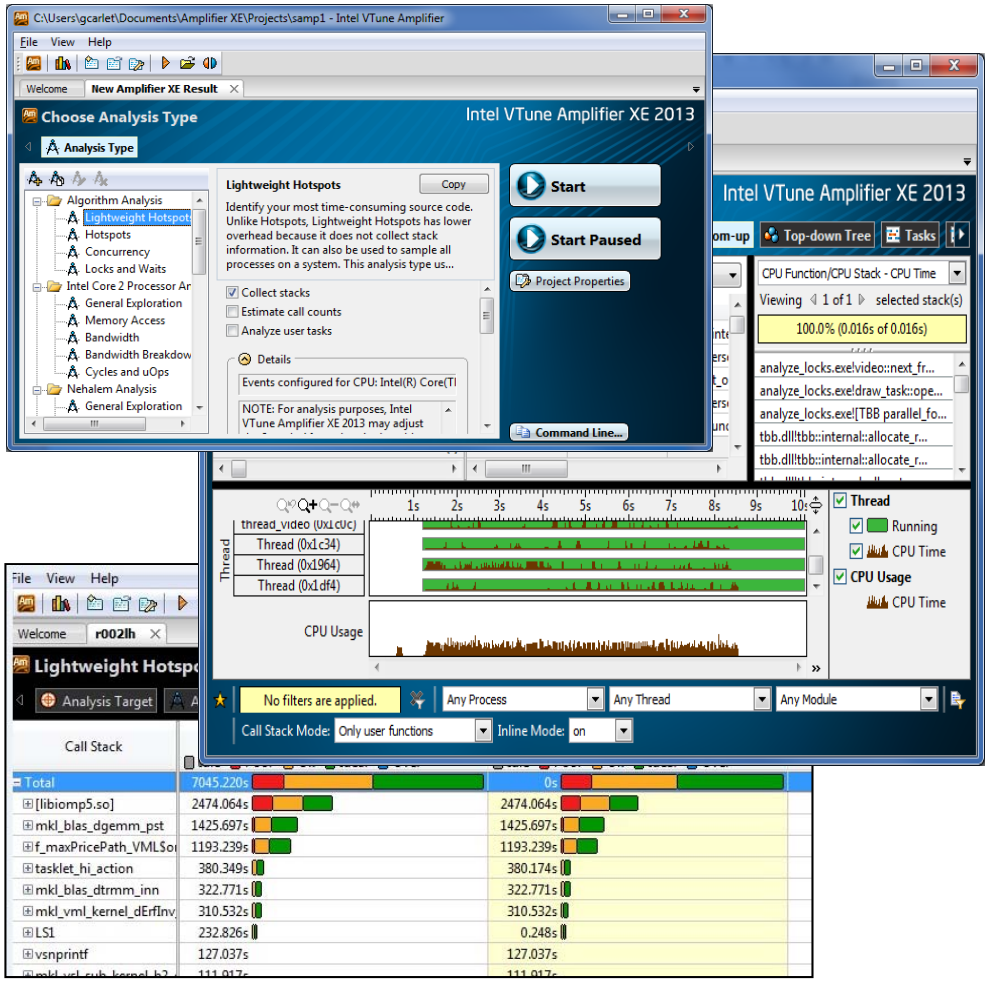
Copyright victor.w.lee@intel.com

Performance Analyzer(s)

- Performance Analyzer:
  - Papi
  - Perf
  - Likwid
  - Gprof
  - Valgrind/kCacheGrind
  - IBM Tivoli

- Other IDEs:
  - Eclipse
  - Microsoft® Visual Studio
  - Xcode from Apple®

- Intel® VTune™ Performance Analyzer



Copyright victor.w.lee@intel.com

Auto-tuner

- Use machine time in place of human time for tuning
- Basic technique is to search over possible implementation
- Recent work start to explore statistical machine learning method to prune search space

Copyright victor.w.lee@intel.com

Auto-tuner Examples

Search of a design space (e.g Kamil'10)

- Start w/ Fortran Spec
- Translate to IR
- Explore design space

Category	Optimization Parameter	Name	Parameter Tuning Range by Architecture Barcelona/Nehalem	Victoria Falls	GTX280
Data Allocation	NUMA Aware		✓	✓	N/A
Domain Decomposition	Core Block Size	*CX*	NX	{8...NX}	{16†..NX}
		CY	{8...NY}	{8...NY}	{16†..NY}
		CZ	{128...NZ}	{128...NZ}	{16†..NZ}
	Thread Block Size	*TX*	CX	CX	$\{1..\frac{CX}{16}\}^{\ddagger}$
		TY	CY	CY	$\{\frac{CY}{16}..CY\}^{\ddagger}$
		TZ	CZ	CZ	$\{\frac{CZ}{16}..CZ\}^{\ddagger}$
	Chunk Size		$\{1...\frac{NX \times NY \times NZ}{CX \times CY \times CZ \times NThreads}\}$		N/A
Low Level	Array Indexing		✓	✓	✓
	Register Block Size	*RX*	{1...8}	{1...8}	1
		RY	{1...2}	{1...2}	1*
		RZ	{1...2}	{1...2}	1*

Other auto-tuners: Examples: ATLAS [Whaley'01], OSKI [Vuduc'05], FLAME [Gunnels '01], FFTW [Frigo'99], SPIRAL [Puschel'05], Stencil Autotuner [Kamil'10] [Datta'08] [Ganapathi'09]

Statistical Machine Learning Model [Ganapathi'09]

- Use Kernel Canonical Correlation Analysis
- Performance within 1% of experts programmers

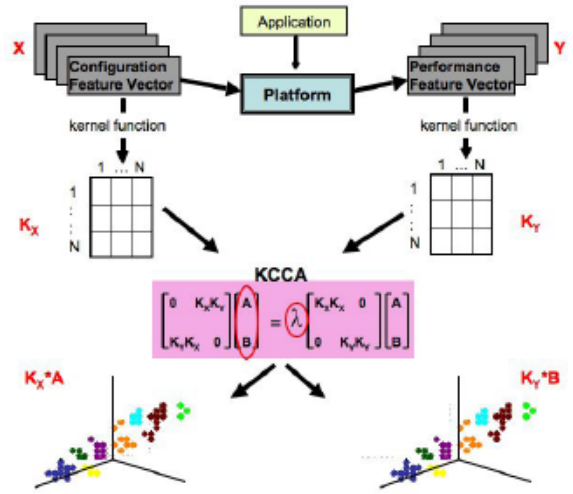
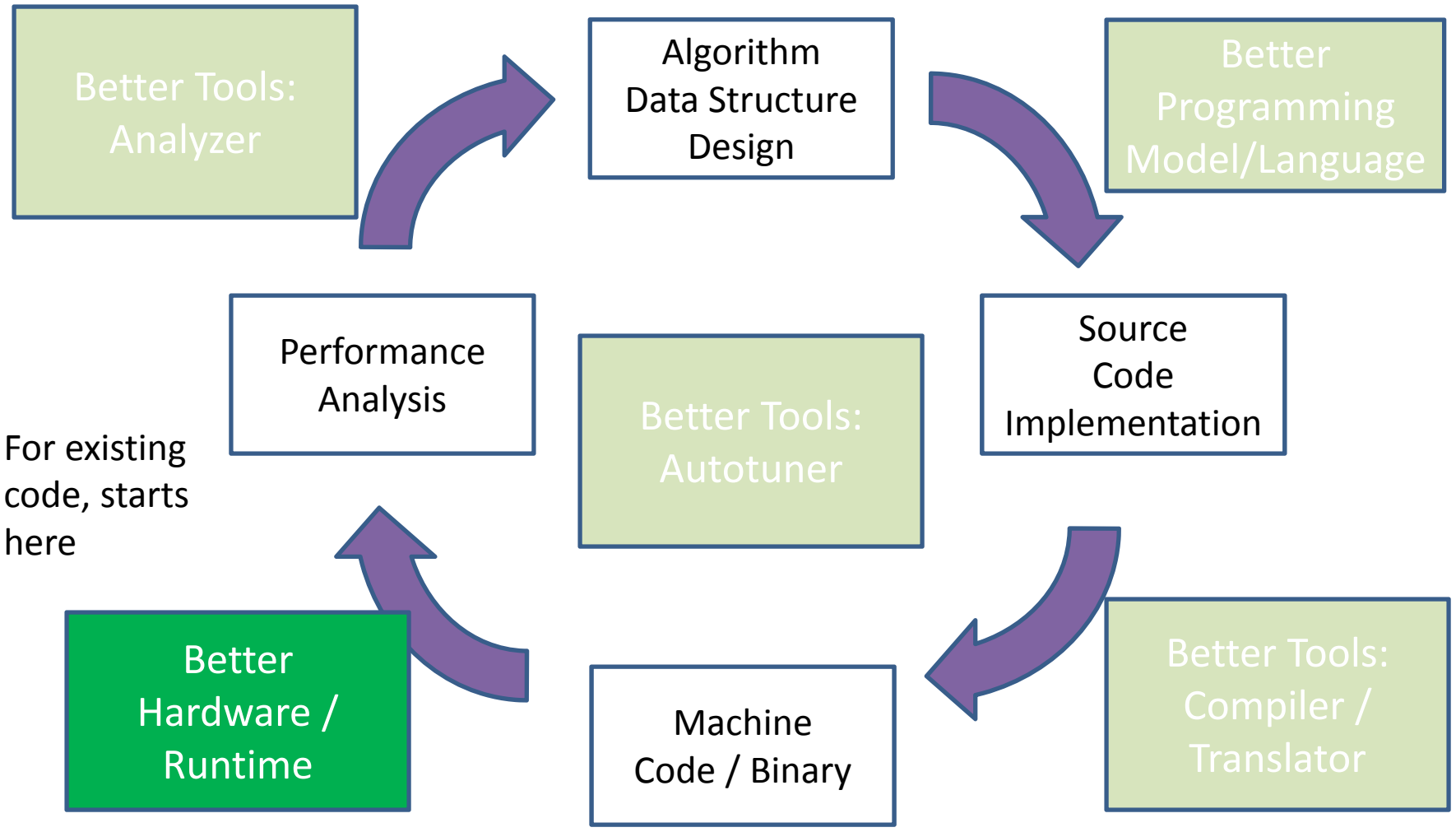


Figure 1: The KCCA methodology discovers relationships between configurations and performance.

Copyright victor.w.lee@intel.com

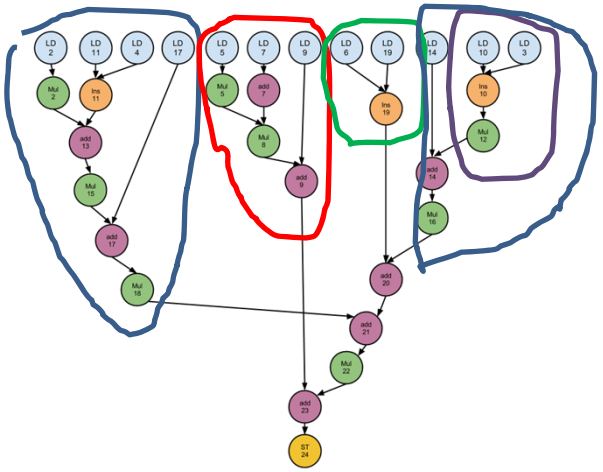
Hardware / Runtime & Perf-Productivity Gap



Copyright victor.w.lee@intel.com

Architecture Supports for Parallelism

- Speculative parallelism
  - Allow multiple iterations or basic blocks on multiple hardware cores/units
  - Examples:
    - Multiscalar [Sohi'95] (parallelize loop iterations)
    - Decouple SW pipeline [Ottoni'05] (parallelize blocks within an iteration)
    - Cluster architecture [Marcuello'99] (group dependent instructions in the same cluster)



Copyright victor.w.lee@intel.com

Architecture Supports for Parallelism

- Managing parallelism
  - Coherent Cache
  - Dynamic load balancing
  - Fast synchronization via. full/empty bit
  - Reduction hardware
  - Transaction memory

Copyright victor.w.lee@intel.com

Architecture Supports for Data Accesses

- Tolerate the latency
  - Multi-threading
- Bring data in earlier (to hide memory latency)
  - Data prefetching
    - Hardware prefetcher, software prefetching
  - Decoupled architecture
    - Helper threads
  - Speculation
    - Value prediction (address prediction [Gonzalez'97])
    - Large Instruction window

Copyright victor.w.lee@intel.com

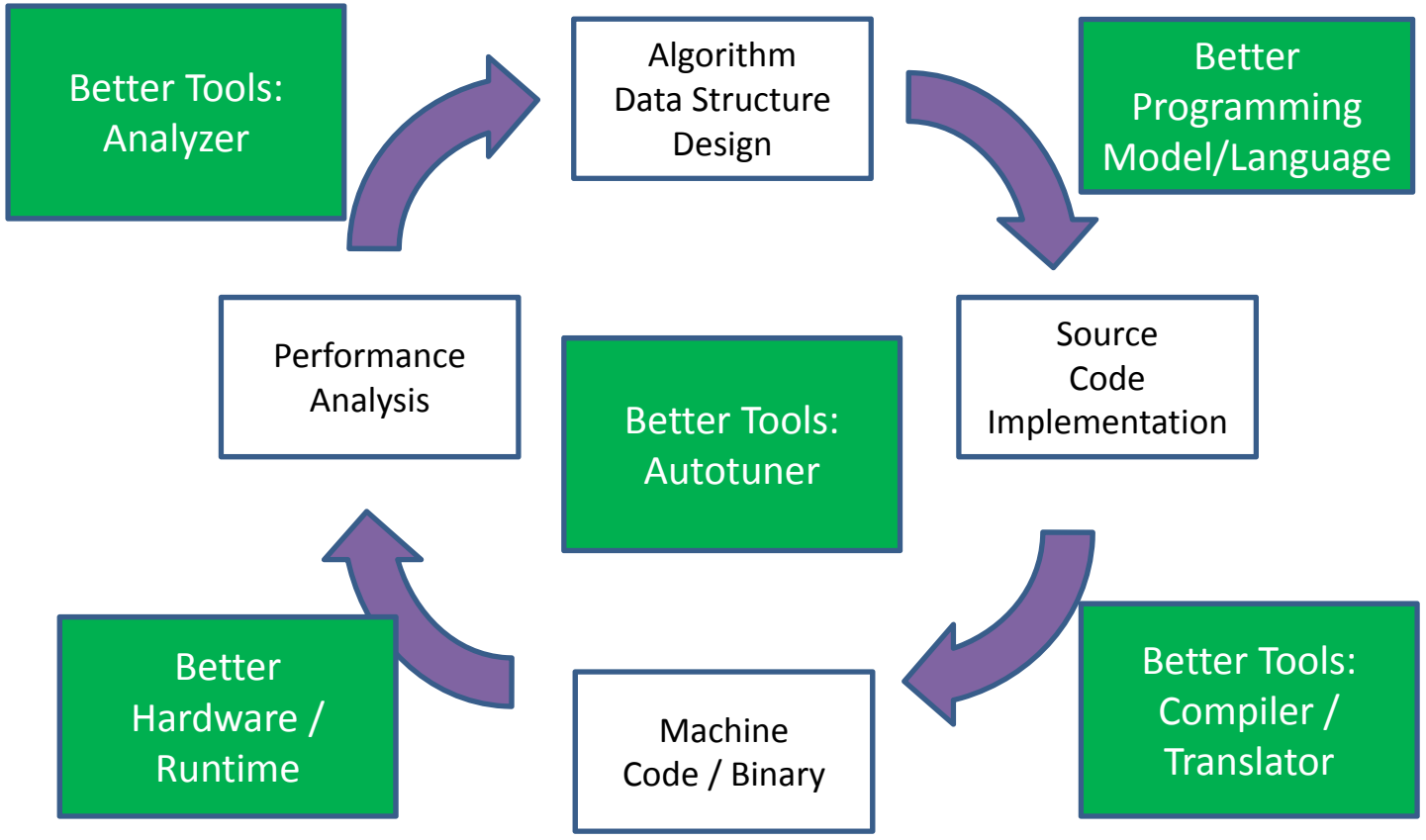
Architecture Supports for Data Accesses

- Conserve the available bandwidth
  - Relax consistency model
    - Allow more overlap between memory requests
  - Write buffers
    - Write buffers or Streaming stores to eliminate unnecessary BW
- Other more extreme idea:
  - Compute in Memory (e.g., IRAM)

Copyright victor.w.lee@intel.com

Mini-Summary 3

- A lot of great researches to build on



Copyright victor.w.lee@intel.com

Agenda

- Performance Productivity Gap
- What Created the Gap
- Past and Current Researches
- Summary

Copyright victor.w.lee@intel.com

Summary

- "Performance Productivity Gap" arises due to a series of technology advances and it leads to competitive disadvantages
- Integrated researches on programming model, language, tools and hardware are need to bridge the "Performance Productivity Gap"
- Many great researches, but more is needed
  - Especially in areas like heterogeneity, energy efficiency, fault tolerance, etc.

Copyright victor.w.lee@intel.com

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Knights Corner, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel, VTune, Xeon, Phi, Cilk, Core, Look Inside and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright ©2013 Intel Corporation.

Legal Disclaimer

Software Source Code Disclaimer: Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

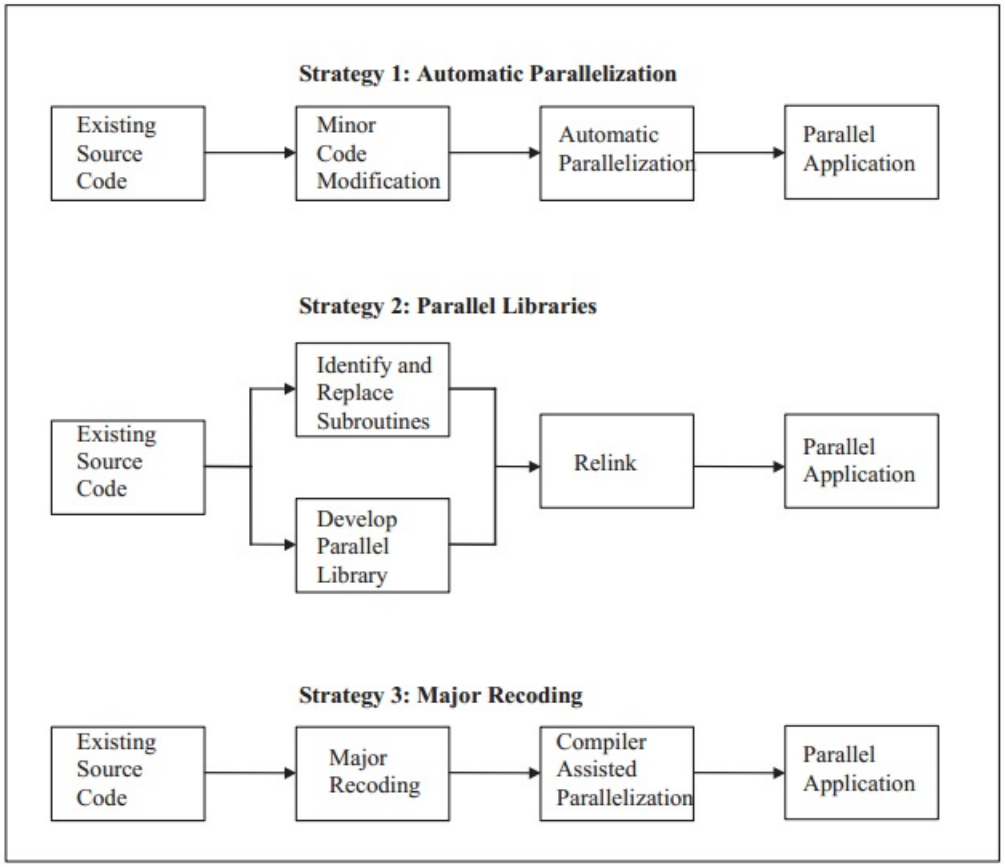
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Backups

Copyright victor.w.lee@intel.com

Parallelization Strategies



Extracted from "High Performance Cluster Computing: Programming and Applications.", by Rajkumar Buyya, Prentice hall PTR, NJ, USA, 1999

Copyright victor.w.lee@intel.com

Reference

General:

[Satish'12] Satish et. al. "Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications?", In proc. of ISCA 2012

[Borkar'13] Shekhar Borkar's IPDPS 2013 Keynote "Exascale Computing – a fact or a fiction?"

[Shivakumar'02] Premkishore Shivakumar, et al. Modeling the Impact of Device and Pipeline Scaling on the Soft Error Rate of Processor Elements. Computer Science Department, University of Texas at Austin, 2002.

Programming Models/Languages

[Barriuso'94] Ray Barriuso and Allan Knies. SHMEM user's guide. Technical report, Cray Inc. Research, May 1994.

[Skjellum'99] Anthony Skjellum, Ewing Lusk, and William Gropp. Using MPI: Portable Parallel Programming with the Message Passing Iinterface. MIT Press, 1999.

[Geist'94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. PVM – Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, 1994.

[Blumofe'95] Robert D. Blumofe, et al. Cilk: An efficient multithreaded runtime system. Vol. 30. No. 8. ACM, 1995.

[CILK'00] CILK-5.3 reference manual. Technical report Supercomputing Technologies Group, June 2000.

[HPF'96] High Performance Fortran Forum. High performance fortran language specification version 2.0. Technical report, Rice University Houston, TX, October 1996.

[Numrich'98] Robert W. Numrich and John Reid. Co-Array Fortran for parallel programming. ACM SIGPLAN Fortran Forum Archive, 17:1–31, August 1998.

[Hilfinger'01] Paul Hilfinger, Dan Bonachea, David Gay, Susan Graham, Ben Liblit, Geoff Pike, and Katherine Yelick. Titanium Language Reference Manual. Technical Report CSD-01-1163, University of California at Berkeley, Berkeley, Ca, USA, 2001.

[El-Ghazawi '03] Tarek El-Ghazawi, William W. Carlson, and Jesse M. Draper. UPC Language Specification v1.1.1, October 2003.

[Carlson'99] W.W. Carlson, D.E. Culler, K.A. Yellick, E. Brooks and K. Warren, "Introduction to UPC and language specification", Center for Computing Sciences Technical Report, CCS-TR-99-157, May 1999.

[Charles'05] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar, "X10: an object-oriented approach to non-uniform cluster computing", 20th annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages and Applications (OOPSLA '05), 2005.

[Allan'05] Eric Allan, David Chase, Victor Luchangco, Jan-Willem Maessen, Sukyoung Ryu, Guy L. Steele Jr., and Sam Tobin-Hochstadt. The Fortress language specification version 0.618. Technical report, Sun Microsystems, April 2005.

[Chamberlain'07] B. L. Chamberlain, D. Callahan, H. P. Zima. "Parallel programmability and the Chapel language". International Journal of High Performance Computing Applications, Vol. 21, No. 3, pp291-312, August 2007.

[OpenMP] OpenMP specifications.

Copyright victor.w.lee@intel.com

Reference

Domain Specific Languages

[Holewinski'12] Justin Holewinski, et al. "High-performance code generation for stencil computations on GPU architectures."Proceedings of the 26th ACM international conference on Supercomputing. ACM, 2012.

[DeVito'11] Zachary DeVito, et al. "Liszt: a domain specific language for building portable mesh-based PDE solvers." Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011.

Compilers (Parallelizing)

SUIF [Wilson'94], Polaris [Blume'94], Helix[Campanoni'12], DSWP[Ottoni'05], Cray compiler [Wichmann'09], ICC [Tian'07]

[Wilson'94] Robert P., Wilson et al. "SUIF: An infrastructure for research on parallelizing and optimizing compilers." ACM Sigplan Notices 29.12 (1994): 31-37.

[Blume'94] William Blume, et al. "Polaris: The next generation in parallelizing compilers." Proceedings of the Seventh Workshop on Languages and Compilers for Parallel Computing. 1994.

[Campanoni'12] Simone Campanoni, et al. "HELIX: Automatic parallelization of irregular programs for chip multiprocessing." Proceedings of the Tenth International Symposium on Code Generation and Optimization. ACM, 2012.

[Ottoni'05] G. Ottoni et al. Automatic thread extraction with decoupled software pipelining. MICRO, 2005.

[Tian'07] Xinmin Tian, et al. "Inside the Intel 10.1 Compilers: New Threadizer and New Vectorizer for Intel Core2 Processors." Intel Technology Journal 11.4 (2007).

[Wichmann'09] Nathan Wichmann, et al. "Early Experience using the Cray Compiling Environment", In Proc. of CUG 2009.

Compilers (Tiling)

[Lim'01] A. Lim, S. Liao, and M. Lam. Blocking and array contraction across arbitrarily nested loops using affine partitioning. In ACM Symposium on Principles and Practice of Parallel Programming, June 2001.

[Rivera'00] G. Rivera and C. Tseng. Tiling optimizations for 3D scientific computations. In Proceedings of SC'00, Dallas, TX, November 2000.

[Sellappa'04] S. Sellappa and S. Chatterjee. Cache-efficient multigrid algorithms. International Journal of High Performance Computing Applications, 18(1):115–133, 2004.

Polyhedral transformations. [Bastoul'04]

[Bastoul'04] Cédric Bastoul, et al. "Putting polyhedral loop transformations to work."Languages and Compilers for Parallel Computing. Springer Berlin Heidelberg, 2004. 209-225.

[Bastoul'04] Cedric Bastoul. Code generation in the polyhedral model is easier than you think. In PACT '04:Parallel Architectures and Compilation Techniques, Washington, DC, 2004.

Copyright victor.w.lee@intel.com

Reference

Auto tuner

ATLAS [Whaley'01], OSKI [Vuduc'05], FLAME [Gunnels '01], FFTW [Frigo'99], SPIRAL [Puschel'05], Stencil Autotuner [Kamil'10] [Datta'08] [Ganapathi'09]

[Whaley'01] R. C. Whaley, A. Petitet, and J. Dongarra, "Automated Empirical Optimization of Software and the ATLAS project," Parallel Computing, vol. 27(1-2), pp. 3–35, 2001.

[Vuduc'05] R. Vuduc, J. Demmel, and K. Yelick, "OSKI: A library of automatically tuned sparse matrix kernels," in Proc. of SciDAC 2005, J. of Physics: Conference Series. Institute of Physics Publishing, June 2005.

[Gunnels '01] John A. Gunnels, et al. "FLAME: Formal linear algebra methods environment."*ACM Transactions on Mathematical Software (TOMS)* 27.4 (2001): 422-455.

[Frigo'99] Matteo Frigo. A fast fourier transform compiler. *SIGPLAN Not.*, 34(5):169–180, 1999.

[Puschel'05] M. P¨uschel, J. Moura, J. Johnson, et al. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, 93(2):232– 275, 2005.

[Kamil'10] Shoaib Kamil, et al. "An auto-tuning framework for parallel multicore stencil computations." *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, 2010.

[Datta'08] Kaushik Datta, et al. "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures." *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008.

[Ganapathi'09] Archana Ganapathi, et al. "A case for machine learning to optimize multicore performance." *First USENIX Workshop on Hot Topics in Parallelism (HotPar'09)*. 2009.

Architecture

[Sohi'95] Gurindar S. Sohi, Scott E. Breach, and T. N. Vijaykumar. "Multiscalar processors." *ACM SIGARCH Computer Architecture News*. Vol. 23. No. 2. ACM, 1995.

[Ottoni'05] Guilherme Ottoni, et al. "Automatic thread extraction with decoupled software pipelining." *Microarchitecture, 2005. MICRO-38. Proceedings. 38th Annual IEEE/ACM International Symposium on*. IEEE, 2005.

[Marcuello'99] Pedro Marcuello, and Antonio González. "Clustered speculative multithreaded processors." *Proceedings of the 13th international conference on Supercomputing*. ACM, 1999.

[Canal'99] Ramon Canal, J-M. Parcerisa, and Antonio Gonzalez. "A cost-effective clustered architecture." *Parallel Architectures and Compilation Techniques, 1999. Proceedings. 1999 International Conference on*. IEEE, 1999.

[Gonzalez'97] J. Gonzalez and A. Gonzalez, "Speculative execution via address prediction and data prefetching", Proceedings of the 1997 International Conference on Supercomputing (ICS-97), 1997

Copyright victor.w.lee@intel.com