CCMT

A Genetic Algorithm based Approach for Multi-objective Hardware/Software Co-optimization

Sanjay Ranka

Computer and Information Science and Engineering

University of Florida PSAAP Center

Outline of the talk

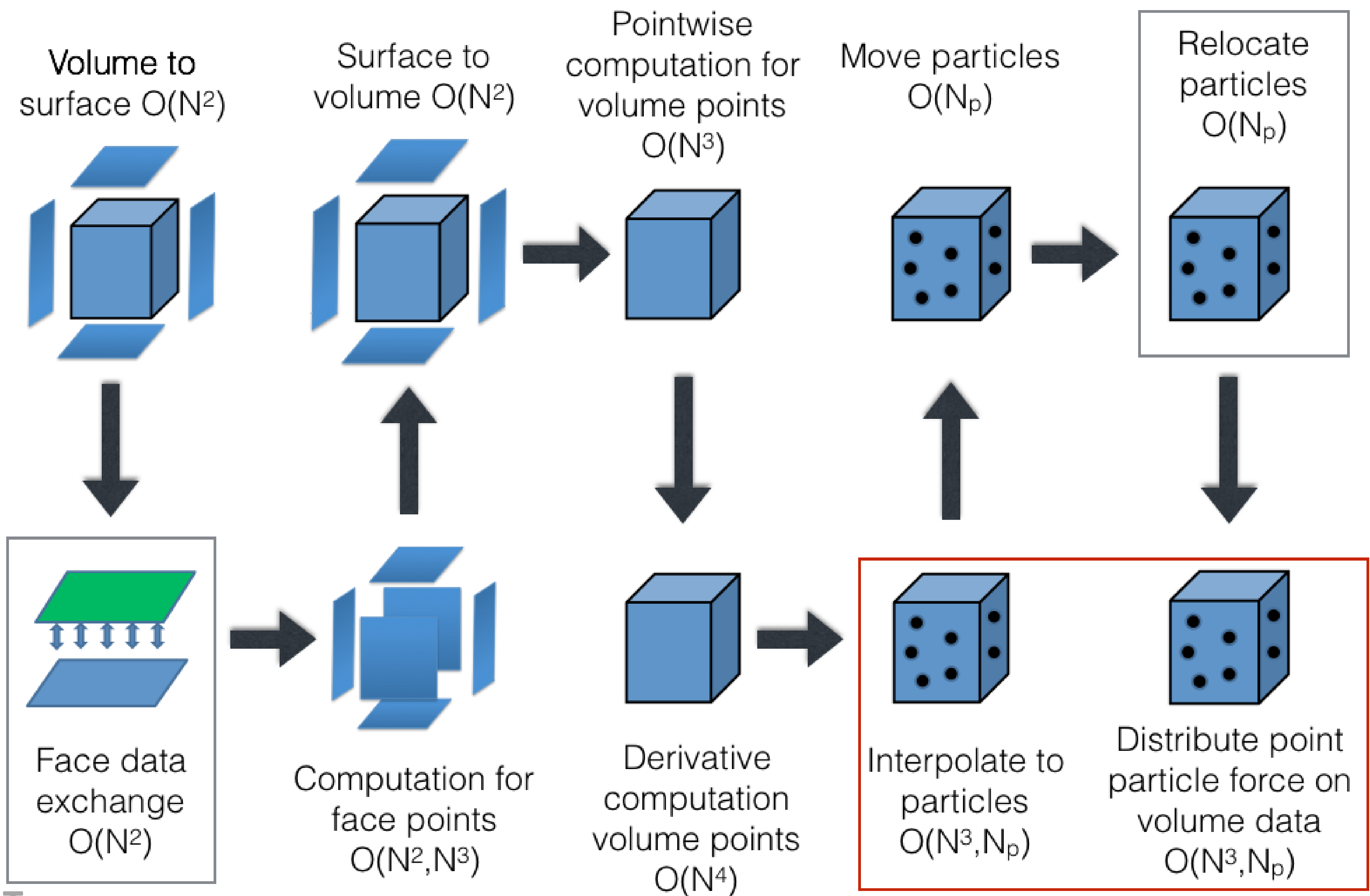
- CMT Bone
- GA based Algorithm for Autotuning
- Experimental Results
- Hybrid Computation
- Conclusions

CMT-bone

- CMT-nek (Compressible Multiphase Turbulence) is the primary simulation tool being developed at the PSAAP-II center at University of Florida to perform state-of-the-art simulations of compressible multiphase turbulent flows
- Leverages NEK5000
- Validation between CMT-Nek and CMT-Bone using Veritas (with LLNL)

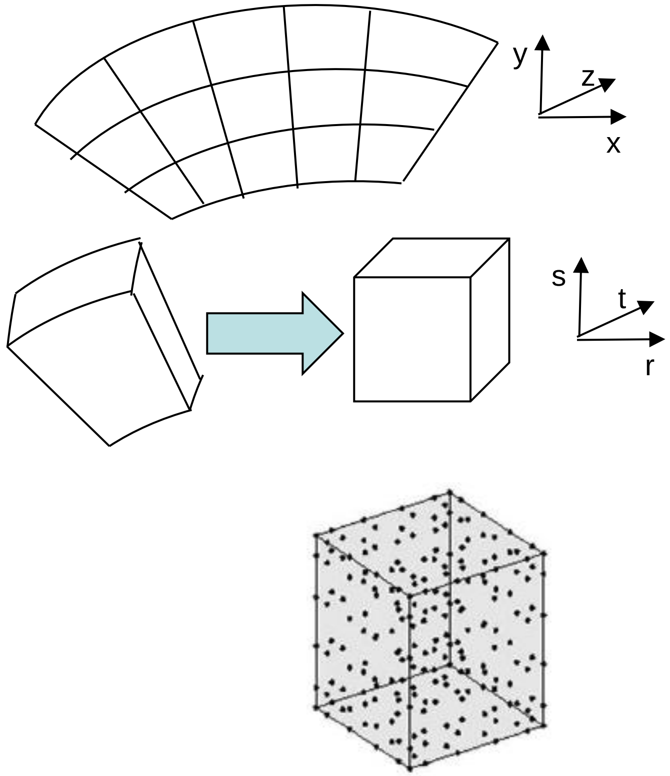
step	CMT-nek subroutine	CMT-nek workflow description	number of variables	
			CMT-nek	CMT-bone
1	`primitive_variables`	*pointwise computation for volume points*	9	5
2	`fillq`	*Volume to surface*	15	10
3	`gs_op`	*face data exchange*	18	10
4	`ausm`	*computation for face points*	5	5
5	`addfacetofull`	*surface to volume*	5	5
6	*evaluate_conv_h*	*Pointwise compuation for volume points*	15	15
7	`flux_div_integral`	*Derivative compuation for volume points*	15	15
8	`baryweights_findpts_eval`	*Interpolate to particles*	3	3
9	`update_stokes_particles`	*Move particles*	3	3
10	`crystal_tuple_transfer`	*Relocate particles*	3	3
11	-	*Distribute point particle force on volume data*	-	-

CMT-bone



Volume to surface $O(N^2)$
Surface to volume $O(N^2)$
Pointwise computation for volume points $O(N^3)$
Move particles $O(N_p)$
Relocate particles $O(N_p)$
Face data exchange $O(N^2)$
Computation for face points $O(N^2,N^3)$
Derivative computation volume points $O(N^4)$
Interpolate to particles $O(N^3,N_p)$
Distribute point particle force on volume data $O(N^3,N_p)$

Derivative Computation Kernel



- $\frac{\partial U}{\partial r}$ (i, j, k) = $\sum_{l=1}^{N_x} A_{il} u_{ljk}$
- $\frac{\partial U}{\partial s}$ (i, j, k) = $\sum_{l=1}^{N_y} B_{il} u_{ilk}$
- $\frac{\partial U}{\partial t}$ (i, j, k) = $\sum_{l=1}^{N_z} C_{il} u_{ijl}$
- If $N_x = N_y = N_z = N$
  - Then $B = C = A^T$
- Complexity: $O(N^4)$
- N is typically between 5-25
  - A large number of small matrix multiplications

The derivative computing kernel requires 25-50% of the total solver time of CMT-nek.

Derivative Kernel

Algorithm: *dudr-4loop*

```
do k = 1, Nz
 do j = 1, Ny
  do i = 1, Nx
   do l = 1, Nx
    dudr(l, j, k) = dudr(l, j, k) +
                    a(i, l) * u(l, j, k, ie)
   enddo
  enddo
 enddo
enddo
```

Algorithm: *dudr-4loop-fused*

```
do k = 1, Nz* Ny
  do i = 1, Nx
    do l = 1, Nx
      dudr(l, k) = dudr(l, k) +
                   a(i, l) * u(l, k, ie)
    enddo
  enddo
enddo
```

Similarly, 5loop versions and 5loop-fused versions were considered.

Algorithm: *dudr-4loop-permuted-and unrolled*

```
do k = 1, Nz
 do i = 1, Nx
  do j = 1, Ny
   do l = 1, Nx,,2
    dudr(l, j, k) = dudr(l, j, k) +
                    a(i, l) * u(l, j, k, ie)
     dudr(l+1, j, k) = dudr(l+1, j, k) +
                    a(i, l+1) * u(l+1, j, k, ie)
   enddo
  enddo
 enddo
enddo
```

Number of 4-loop implementations for dudr $N_x=N_y=N_z=10$

$= 4! * 4 \wedge 4$

$= 24 * 256 = 6144$ variants

Total number of variants = 98,240 (N=10)

Total number of variants = 217,728 (N=20)

Exhaustive search may not be feasible

Related Work: C. Chen, J. Chame, M.W. Hall, CHiLL: A Framework for Composing High-Level Loop Transformations, Technical Report 08-897, University of Southern California, Computer Science Department, 2008.

Genetic Algorithm

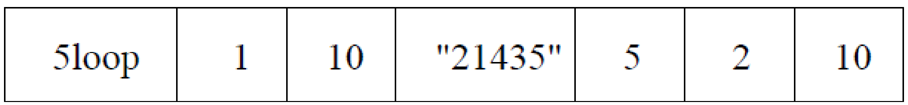


Figure 9. Example of an individual

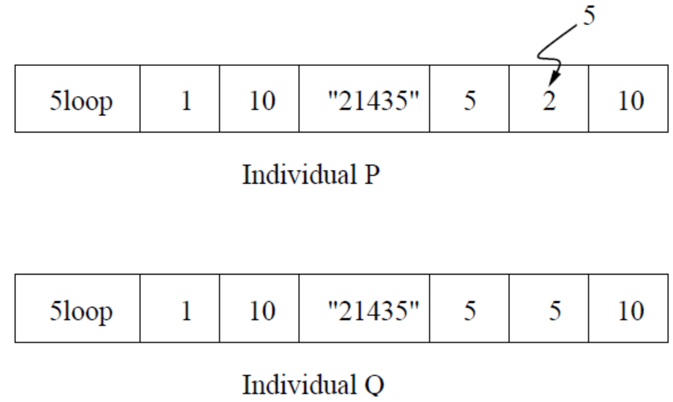


Figure 11. Mutation applied to individual P

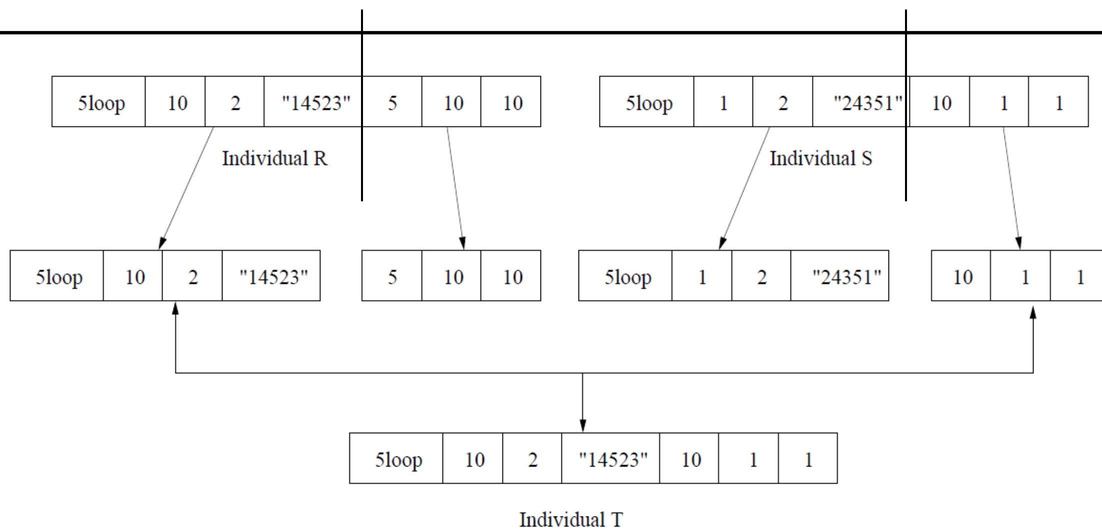
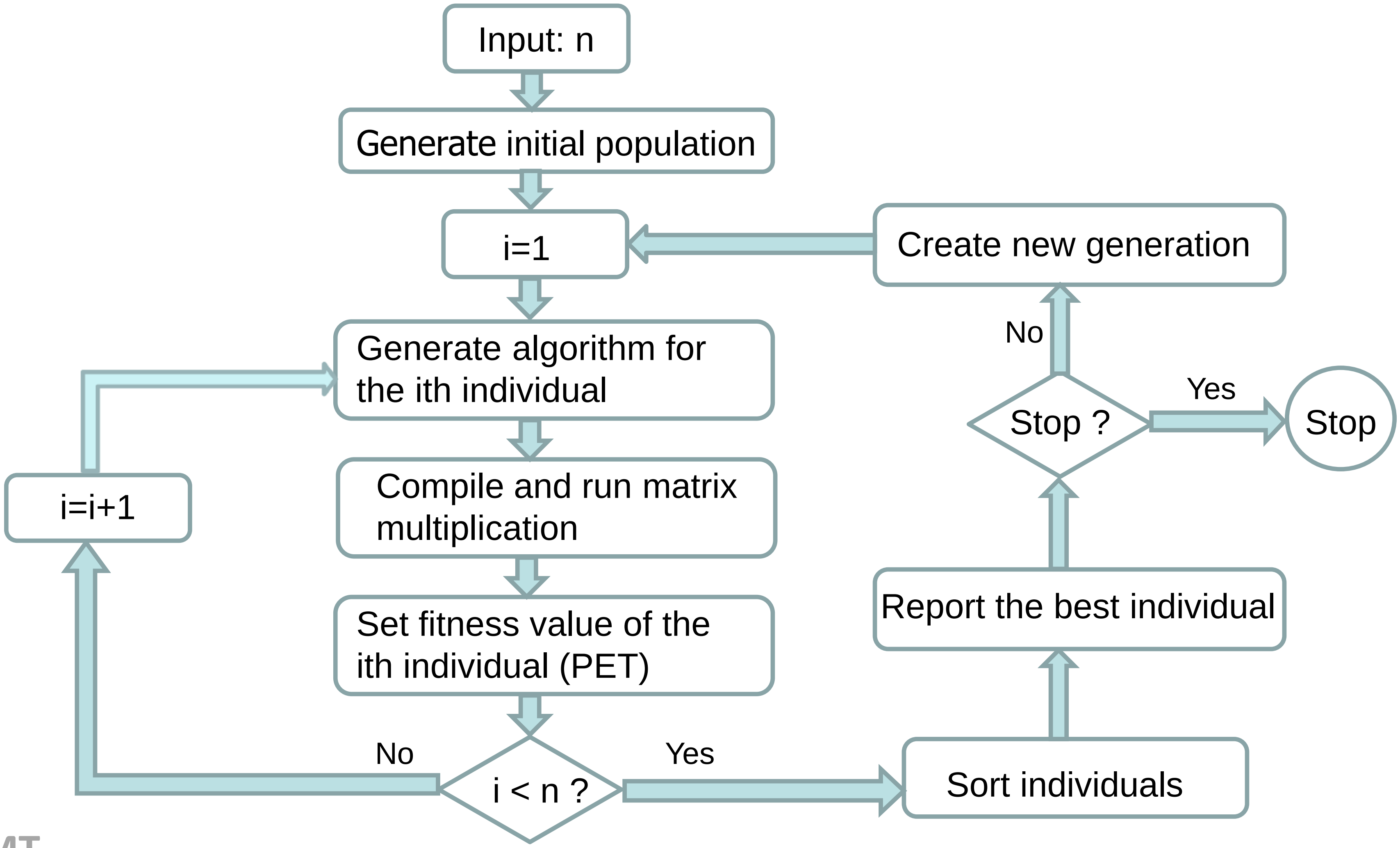


Figure 12. Crossover operator applied to individuals R and S

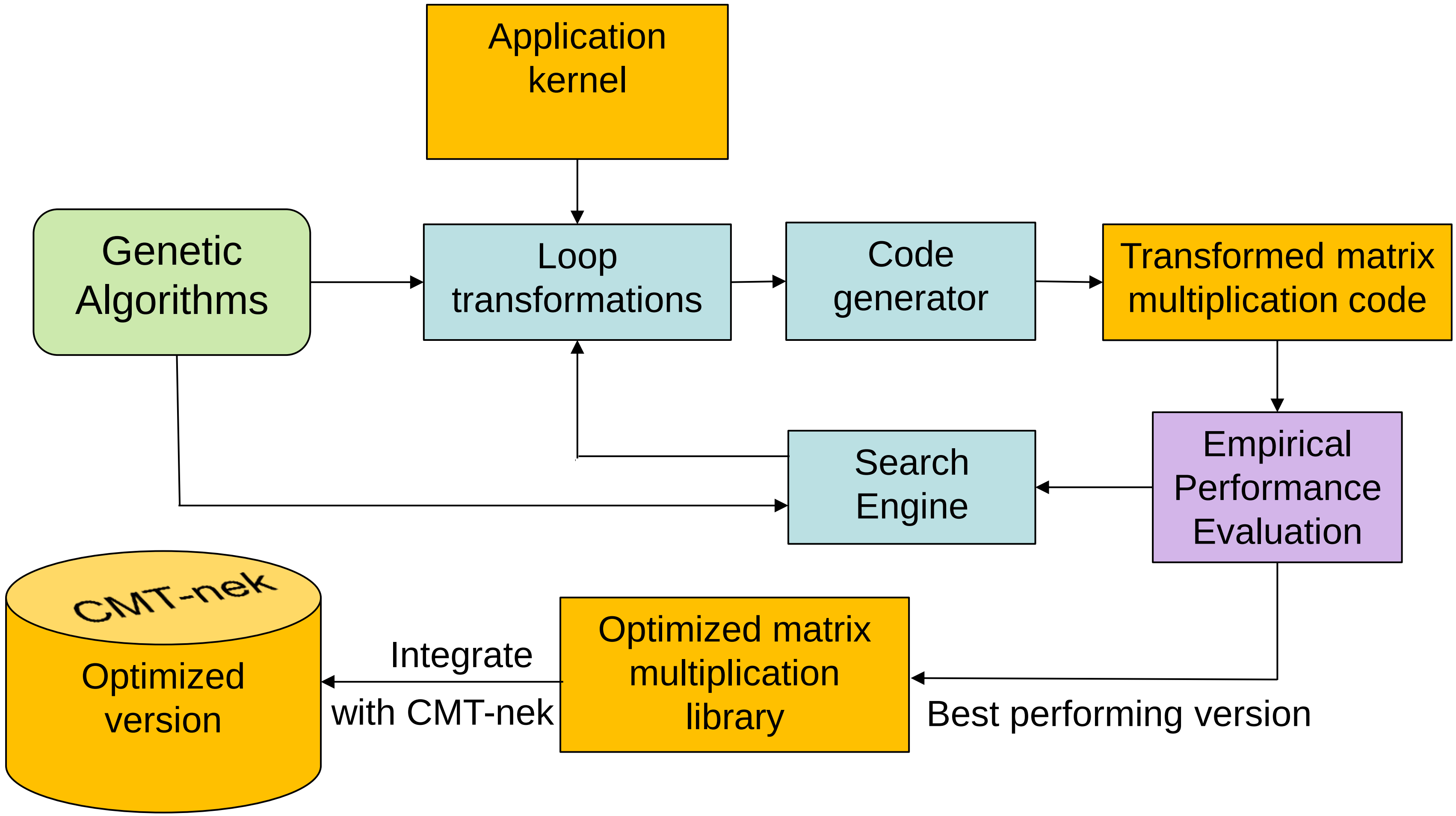
CCMT

Genetic Algorithm

- We use genetic algorithms to search the exploration space efficiently.
- Individuals represent matrix multiplication variants



Autotuning Framework



Performance And Energy

- Software Implementation:
  - CMT-nek
  - 4loop version
  - 4loop-fused version
  - 5loop-version
  - 5loop-fused version

- CPU Platforms:
  - IBM Blue Gene/Q
  - AMD Opteron 6378
  - AMD Fusion

- BG/Q node
- Cores: 16
- Each core:
  - 4-way SMT
  - 1.6 GHz
- 204.8 GFLOPS peak performance
- 55W peak power

- Dell 6145 node
- 4 AMD Opteron CPU
- Each CPU:
  - 16 cores
  - 2.4 GHz
- 614.4 GFLOPS peak performance
- 115 W peak power

- AMD Fusion
- 104 nodes cluster
- Each CPU:
  - 4 cores
  - 3.8 GHz

Comparison of GA with Exhaustive Approach

- Comparison of performance by platform

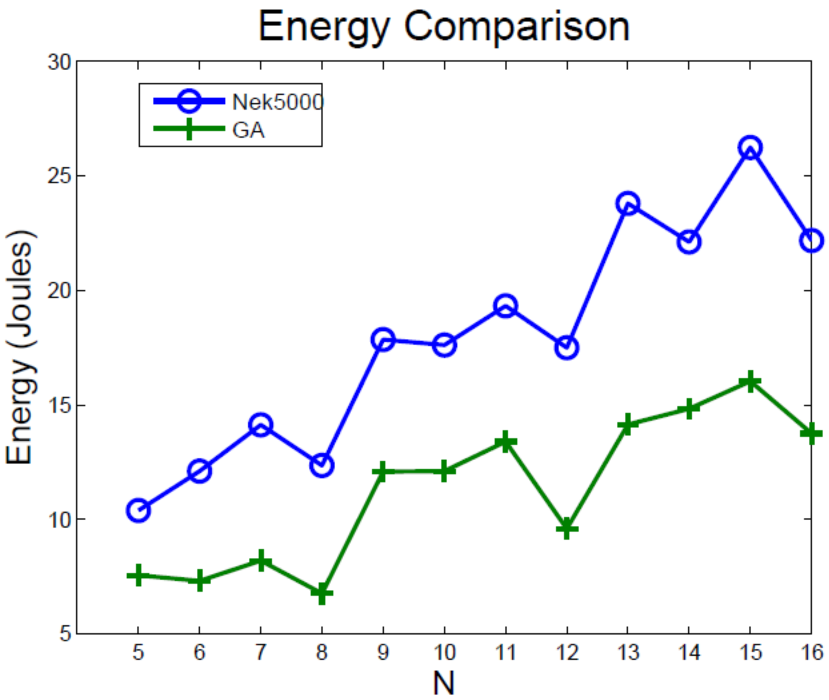
Platforms	CMT-nek time (seconds)	Exhaustive Autotuning			GA based Autotuning			
		Time	Variants	%impro-ve(cmt)	Time	Variants	%impro-ve(cmt)	%lowerPerf
IBM BG/Q	5.57	2.54	97716	54.3	2.58	1124	53.7	1.1
AMD Opteron	1.81	1.02	97716	43.6	1.06	1283	41.4	3.9
AMD Fusion	1.36	0.95	32652	30	0.95	485	30	0.0

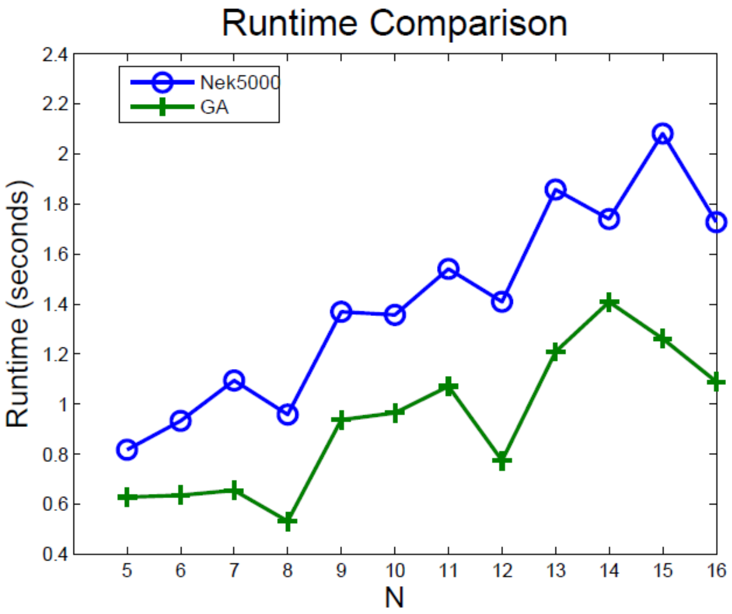
- Comparison of energy consumption by platform

Platforms	CMT-nek energy (Joules)	Exhaustive Autotuning			GA based Autotuning			
		Energy	Variants	%impro-ve(cmt)	Energy	Varian ts	%impro-ve(cmt)	%lowerEnergy
IBM BG/Q	292.1	131.7	96 top	54.9	135	96 top	53.7	2.5
AMD Fusion	17.6	11.98	32652	32	12.11	485	31	1.1

UF UNIVERSITY of FLORIDA

Results (AMD Fusion @ Sandia)

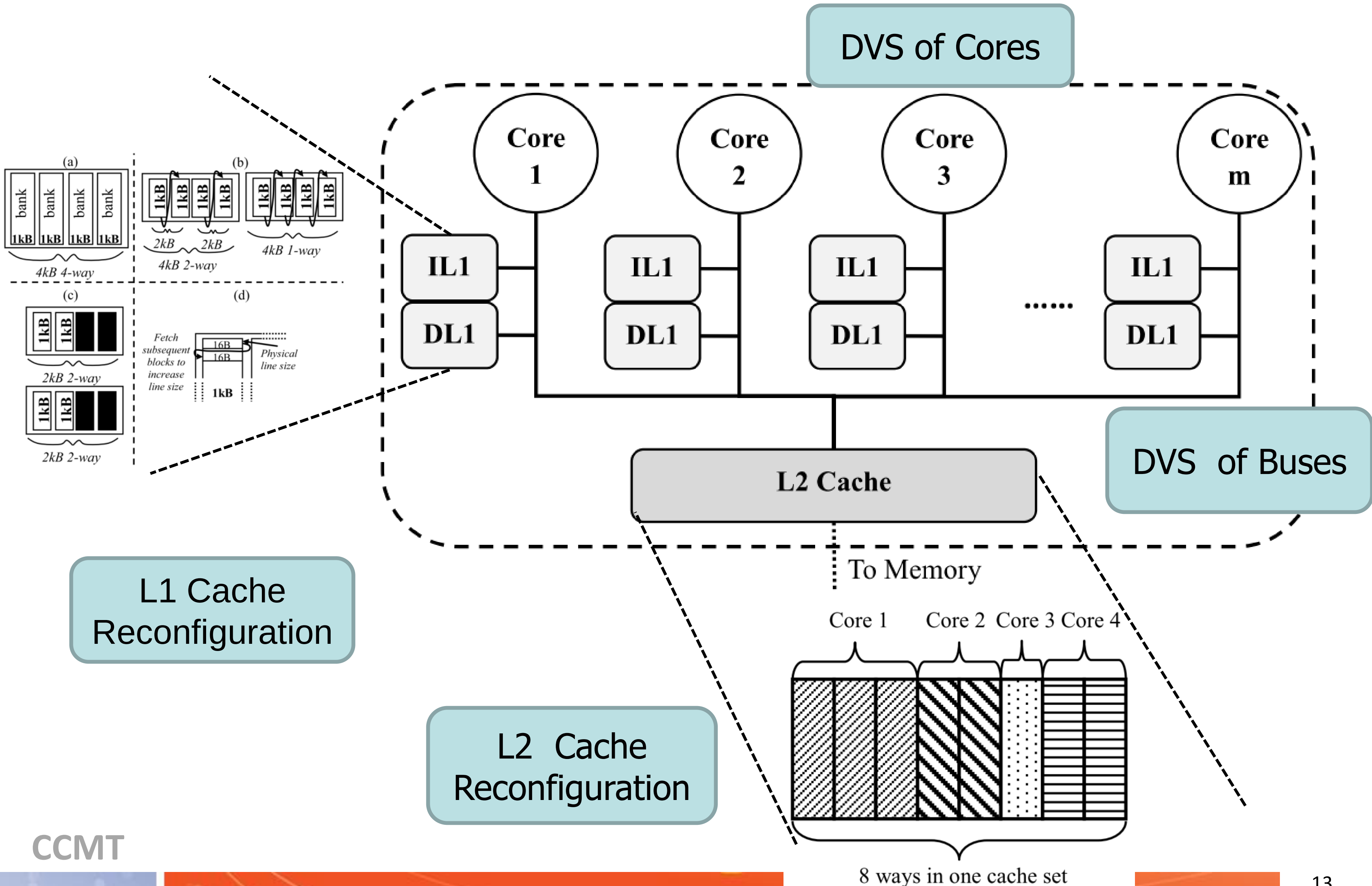




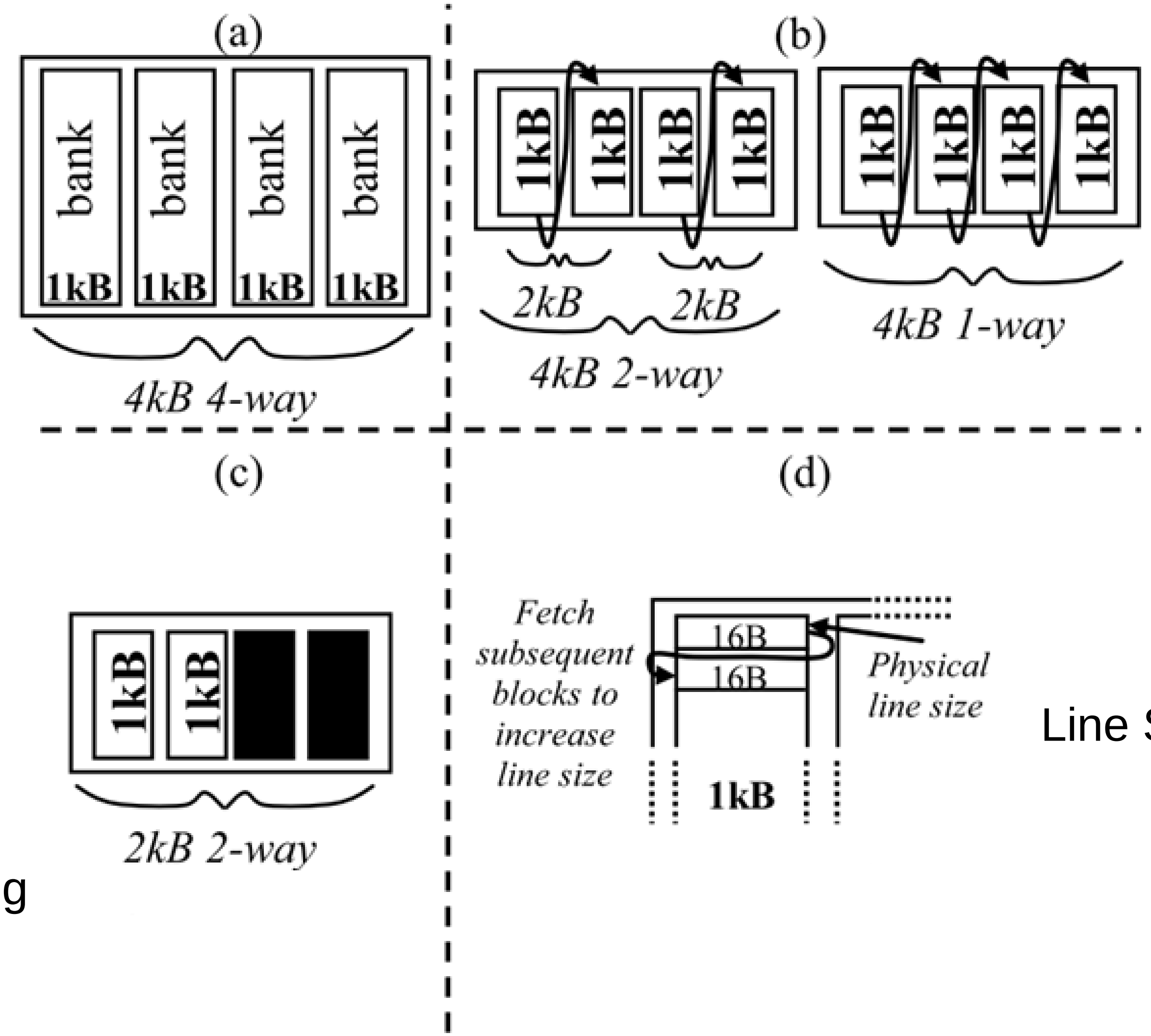
- Between 27% to 45% improvement average improvement of 37%
- Maximum improvement for N=8 and N=12

- Between 23% to 45% improvement, average improvement of 34%
- Average power consumption is about the same for the various implementations across different matrix sizes
- Improvement in energy consumption heavily reflects improvement in runtime

Optimizing Hardware Configurations



Reconfigurable Cache



Associativity Tuning

Line Size Tuning

Capacity Tuning

Zhang et al., ACM TECS 2005

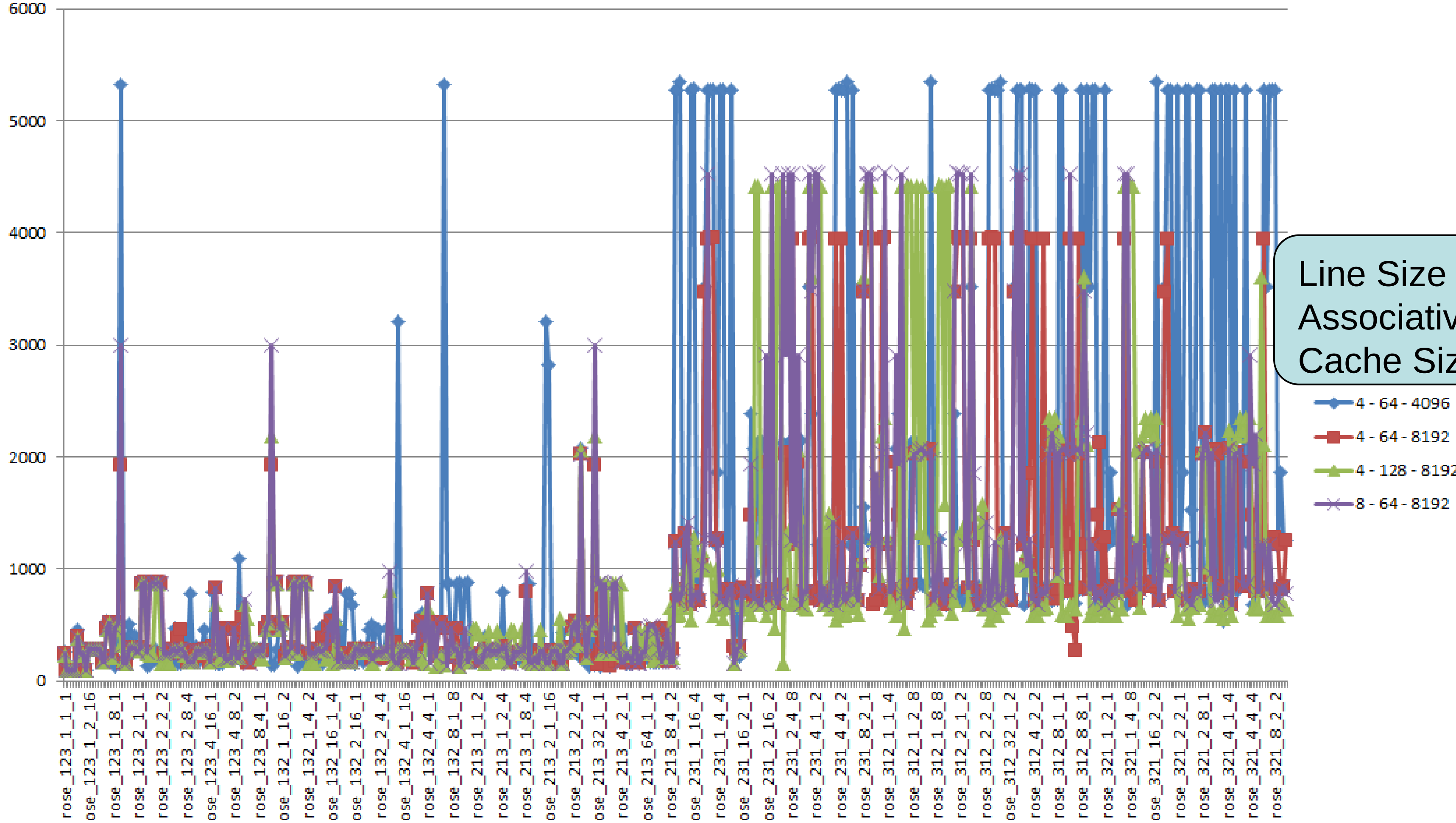
Hardware Software Co-optimization

- Hardware parameters
  - L1 cache size – 2K, 4K, 8K
  - L1 line size – 64, 128, 256
  - L1 associativity – 2, 4, 8
- Code Information
  - Partial derivative computation along direction r
- Software parameters
  - Loop permutation
  - Loop unroll factors
- Problem Size (N) – 16
- Number of Code Variations – 4500
- Details of the GEM5 environment
  - Instruction set architecture: X86
  - CPU model: Out-of-order CPU
  - Memory model: Classic, DDR3
  - Clock frequency: 1GHz

UF UNIVERSITY of FLORIDA

Variation in Time

Time (microseconds)

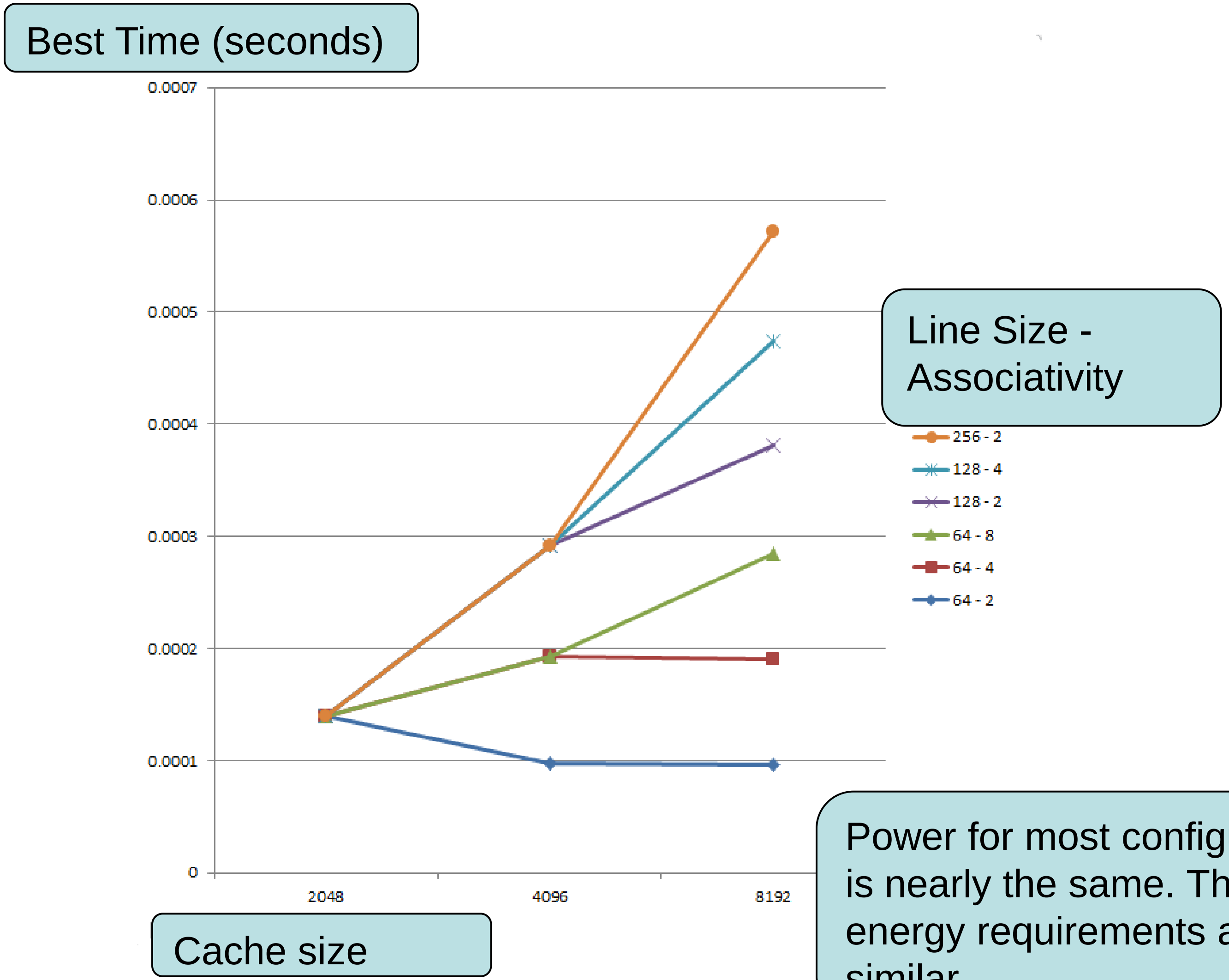


Line Size – Associativity - Cache Size

Code Variation

CCMT

Performance (varying hardware parameters)

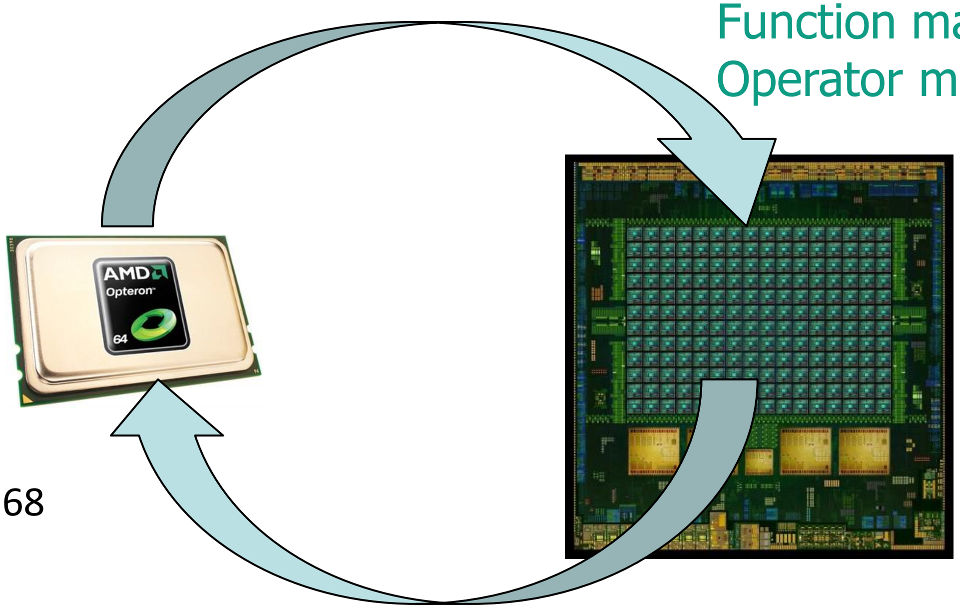


Power for most configurations is nearly the same. Thus, energy requirements are similar.

Hybrid implementation of CMT-bone

- Master-slave
- CPU sends function and operator matrices to GPU, GPU computes derivatives and sends back to CPU

Function matrices,
Operator matrices



Idle cores utilized to compute derivatives.

Tesla K20c:
- 13 Processors
- 192 Cores
- 48k shared memory
- 64k registers
- 1170 GFLOP/s Peak
- 706MHz clock frequency

Host:
- AMD Opteron 6168
- 12 cores
- 1.9GHz clock frequency

derivatives

Optimization Approaches

- On GPU:
  - The derivative operator matrices are only brought in once per block from the device memory to the shared memory. This reduces the number of memory transactions required.
  - The derivative operator matrices are stored in registers instead of shared memory. This reduces the number of accesses to the shared memory.
- On CPU:
  - Using code transformation with proper loop unroll factor and permutation using CHiLL
- Load balancing strategies on CPU and GPU
  - Performance optimal and energy optimal strategies

Modeling Runtime, Power and Energy

GPU	CPU
Runtime $Tgpu = Tcomp + Tcomm$ $Tcomp = 7.17 \times 10^{-11} \times N^{3.76} \times Y$ $Tcomm = 6.14 \times 10^{-9} \times N^{3} \times Y$	Runtime $Tcpu = 1.02 \times 10^{-9} \times N^{4.4} \times Y$
Power $P_{GPU} = 162.24 \times N^{-0.1}$ Nearly a constant, decreasing very slightly with increasing N. Number of memory transactions per unit of data use decreases with N.	Power $P = Pmemory + Pcore$ $= 5.95 + 12.21 \times N^{0.3}$
Energy $E_{GPU} = 9.28 \times 10^{-7} \times N^{3} \times Y$	Energy $E_{GPU} = 2.51 \times 10^{-8} \times N^{4.5} \times Y$
Runtime on *g* GPUs $Tggpu = 1/g * T\ gpu$	Runtime on *p* CPU cores $Tpgpu = 1/p * Tcpu$

Load Balancing Results

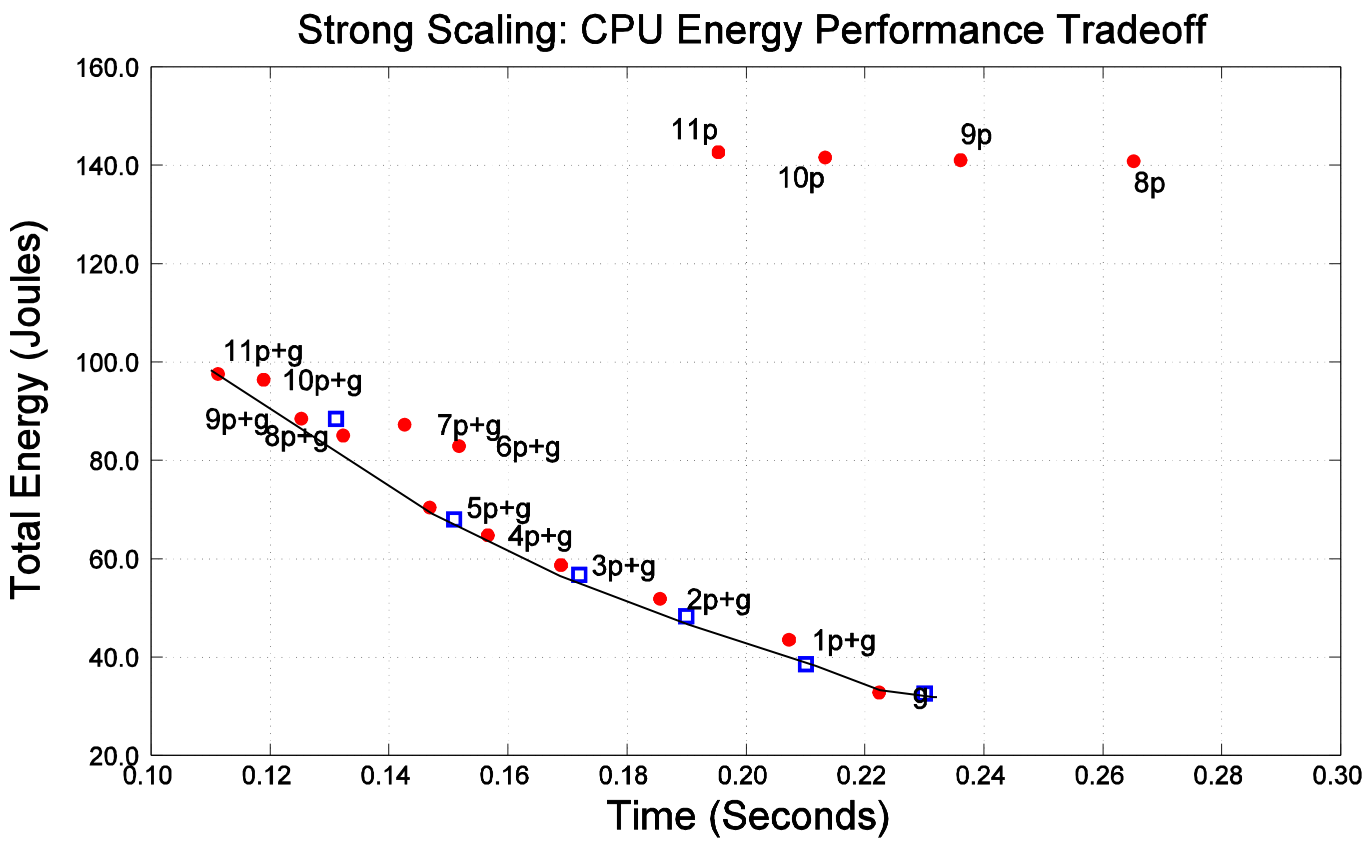
- Optimal Energy: Given a deadline, the GPU should process most of the load with the remaining load being processed by the CPU

Deadline(s)	Time(s)	Power(W)	Energy(J)	GPU(%)
0.210	0.210	213	38.6	94.46
0.190	0.189	279	48.37	85.47
0.126	0.126	741	93.22	44.56

- Optimal Performance: CPU and GPU finish processing at about the same time

Deadline(s)	Time(s)	Power(W)	Energy(J)	GPU(%)
0.210	0.2072	213	43.5	91.25
0.190	0.185	279	51.8	82.83
0.126	0.1189	807	95.17	43.75

Energy Performance Tradeoff

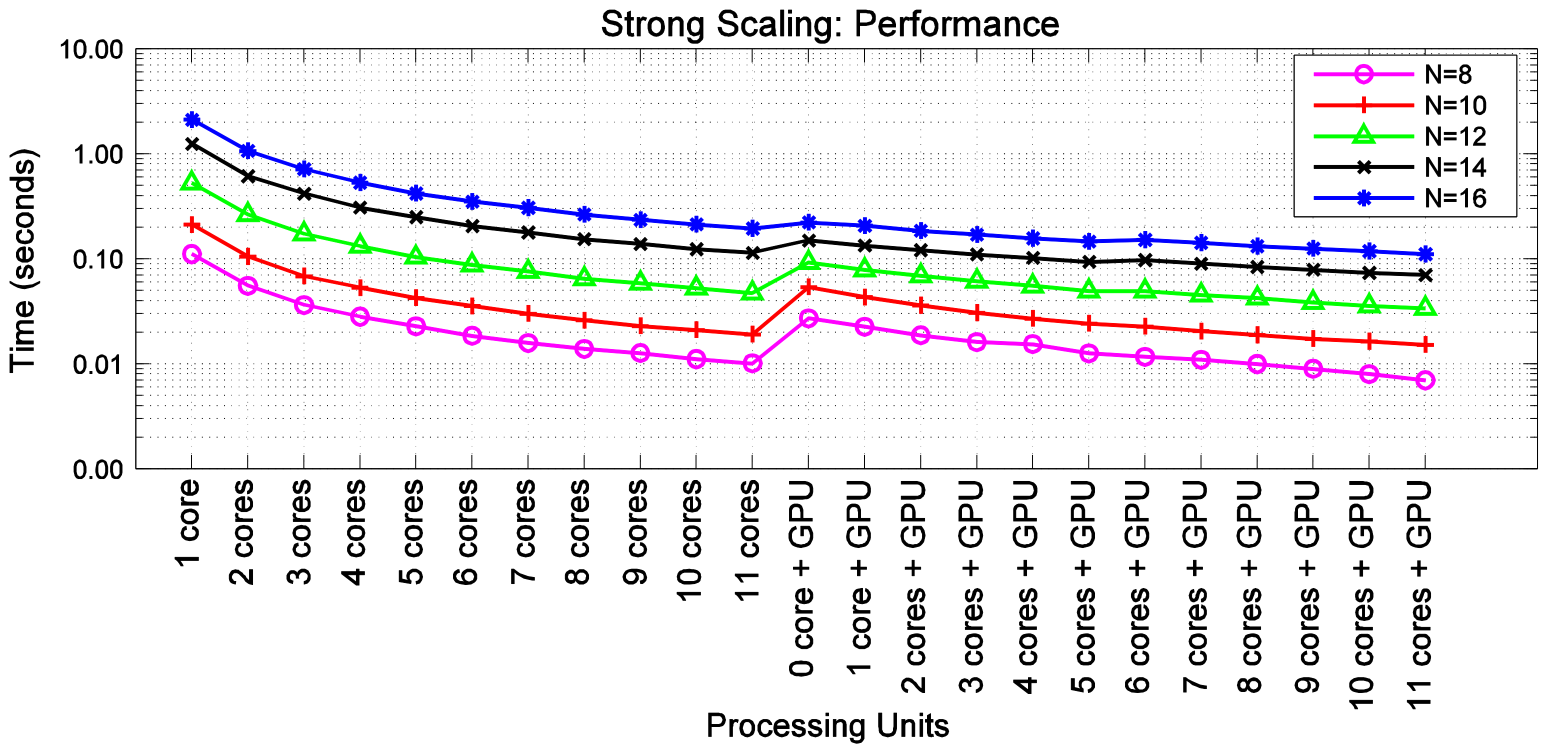


Circles represent performance and power for a performance optimal implementation

Squares represent performance and power for an energy optimal implementation

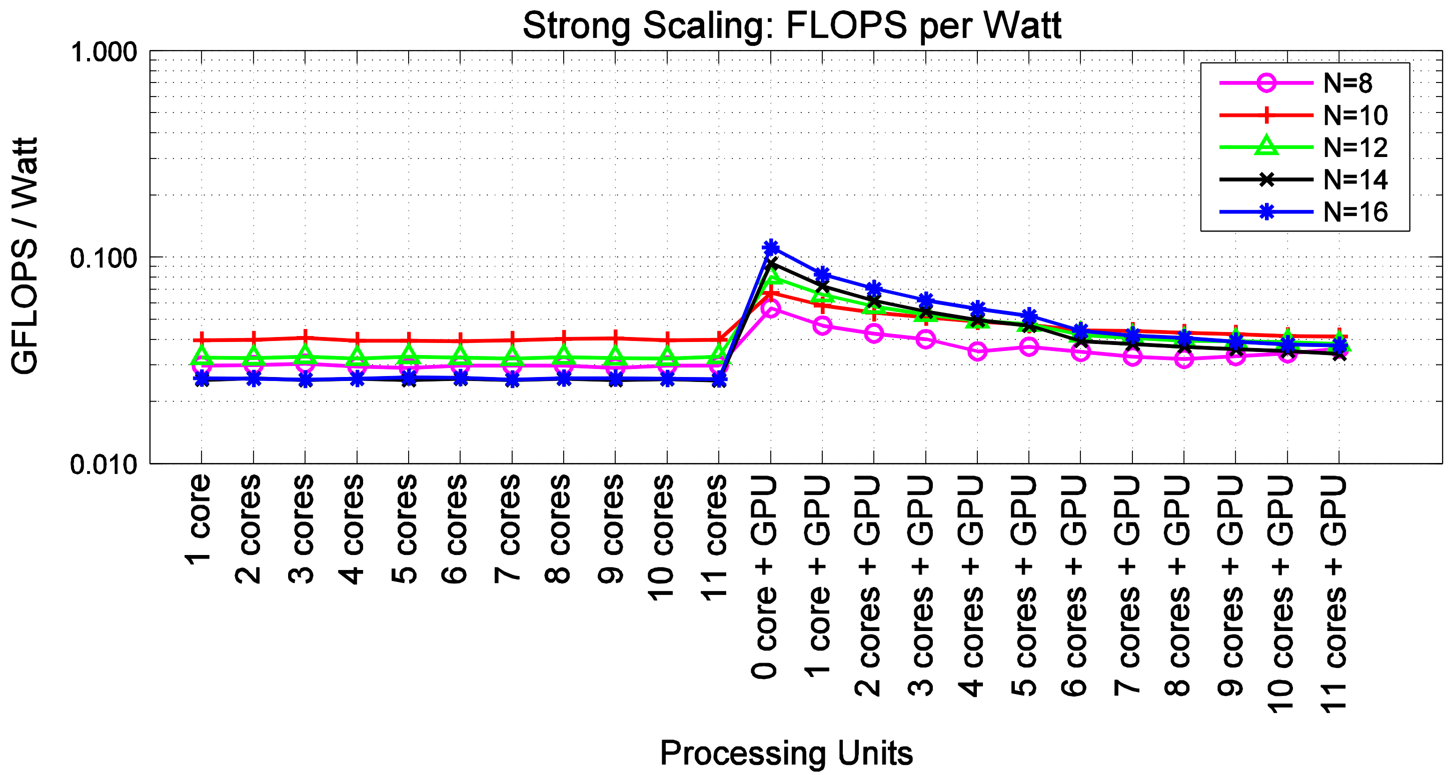
Pareto optimal front for hybrid architectures

CPU + GPU Performance



- Load = 10000 spectral elements
- For CPU-only configurations, runtime drops by a factor of the number of cores
- For CPU-GPU configurations, runtime does not drop as profoundly

CPU + GPU Power Performance



- CPUs are efficient for smaller matrices
- GPUs are efficient for larger matrices
- CPU+GPU efficiency tend to be independent of matrix size as more CPU cores are added

Conclusions

- Developing CMT-bone as a proxy app for CMT-nek
- GA based Autotuning as an important strategy for improving both performance and energy
  - We achieved between 23-61% improvement in performance and about 27-55% improvement in energy requirement
  - Only 1.49% of search space was explored at most to obtain near- optimal results. Performance and energy consumption were within 4% of optimal result.
- Developed novel methods for optimizing CMT kernels on hybrid processors
  - Performance Optimal or Energy Optimal require different approaches

Genetic Algorithm based Autotuning Approach for Performance and Energy Optimization, Tania Banerjee and Sanjay Ranka, IGSC, 2015.

Multiobjective Optimization of Spectral Solvers of Hybrid Multicore Platforms, Tania Banerjee, Jacob Rabb and Sanjay Ranka, submitted to SUSCOM.

CMT-bone -- A Proxy Application for Compressible Multiphase Turbulent Flows Jason Hackl, Mrugesh Sringarpure, Tania Banerjee, Tanzima Islam, S. Balachandar,Thomas Jackson, Sanjay Ranka, in preparation.