Performance Evaluation of Cache Optimized CFD Codes Applied to Complex Flow Applications

Raymond P. LeBeau, Jr, Saurabh Gupta,
Anand Palki
University of Kentucky

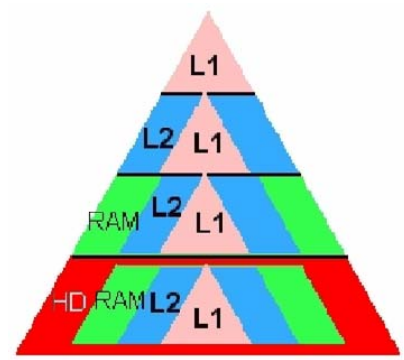
Thanks to Kentucky NASA EPSCoR and the Kentucky Science and Engineering Foundation

Motivation

- Kentucky NASA EPSCoR Grant
- Develop and apply techniques to improve performance and efficiency of CFD codes
- Target commodity cluster architectures
  - Single processor performance
  - Multiple processor performance
- Evaluate changing architectures
- Educate and involve students

CFD Code Optimization

- Optimization includes single processor, multi-processor, and I/O performance
- Single node performance is strongly influenced by cache performance.
- Techniques
  - – Cleaning
  - – Data Restructuring
  - – Loop Restructuring
  - – Algorithm restructuring
  - – TLB
  - – Blocking



CFD Code Optimization

- Stages of Single Processor Optimization
  1. Development and testing of code performance enhancements on core code for simple flows
  2. Testing of code performance across multiple architectures, compilers, problem sizes
  3. Extension of improvements to other aspects of code
  4. Testing of enhanced codes on increasingly complex flows
  5. User-ready version of the code
  6. Implementation of best improvements in more complex CFD codes.

GHOST

- In-house incompressible CFD code
- Finite-volume, structured, 2D
- 2nd order accuracy in time and space
- Implicit - sub-iteration, tri-diagonal solver
- Advective terms – QUICK
- RANS turbulence models - TVD
- Diffusive terms – Central difference
- Pressure – Rhie and Chow
- Original data stored in individual $\phi(i,j)$ arrays

UNCLE

- In-house incompressible CFD code
- Finite-volume, unstructured, 2D/3D
- 2nd order accuracy in time and space
- Implicit - sub-iteration, gaussian-point solver
- Advection terms – Upwind scheme
- Diffusive terms – Central difference
- Pressure – SIMPLE, Rhie and Chow
- Data stored as vectors of structures corres-ponding to the cells, faces, and vertices:

$\Phi(i):\phi_1,\phi_2,\ldots$

Kentucky Fluid Clusters

KFC4
- 47 nodes
- AMD 2500+, 32-bit, 1.83 GHz, 512 KB L2
- 512 MB RAM
- Single-switch 100Mb and single-switch 1Gb network

KFC5
- 47 nodes
- AMD64 3200+, 64-bit, 2.01 GHz, 512 KB L2
- 512 MB RAM
- Single-switch Gb network

Kentucky Fluid Clusters

KFC6I
- 24 nodes
- Intel E6400 dual core 64-bit, 2.13 GHz, 2 MB L2
- 1 GB RAM
- Single-switch Gb ethernet

KFC6A
- 23 nodes
- AMD64 4600+ dual core 64-bit, 2.40 GHz, 2x512 KB L2
- 1 GB RAM
- Single-switch Gb ethernet

Tools

Valgrind

- Version 3.1
- Cachegrind tool
- Cache simulation software
- Instruction & Data
- Miss rates for L1, L2
- http://valgrind.kde.org

Intel - compiler

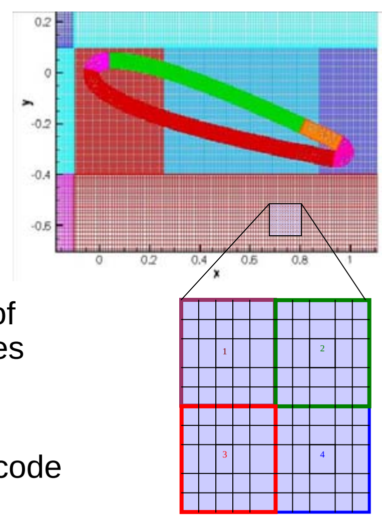
- Version 9.0
- -openmp -fpp -ipo -O3 -no-prec-div -static -ip -pad -Vaxlib –w
- Problems with stack size limitations

Definitions

- Normalized walltime = wallime/iteration/gridpoint (μs)
- Cache miss rate = cache misses/calls

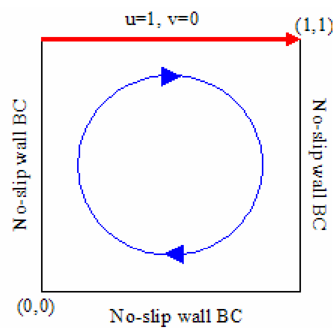
GHOST Optimization

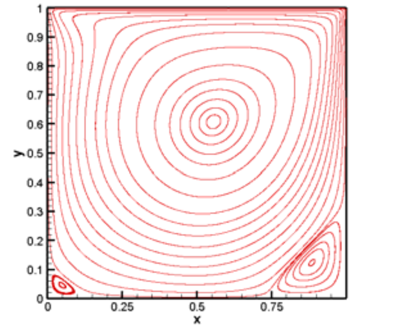
- Verison 0 (V0) - Original code
- Version 1 - Loop correction, Code cleaning
- Version 2 - Loop restructuring
- Version 3 (V3) - Data restructuring
- External Blocking - Blocking at grid generation
- Turbulence (V0t, V3t) - Extension of improvements to turbulence routines
- Unsteady - Incorporation of steady subiteration into unsteady code
- Internal Blocking - Blocking within code

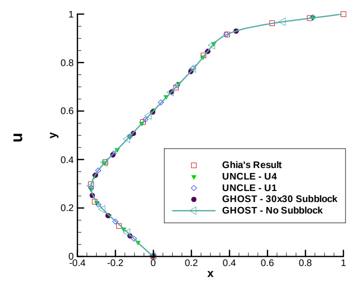


Base Test Problem

Driven flow in a square cavity
Uniform grid
Steady flow

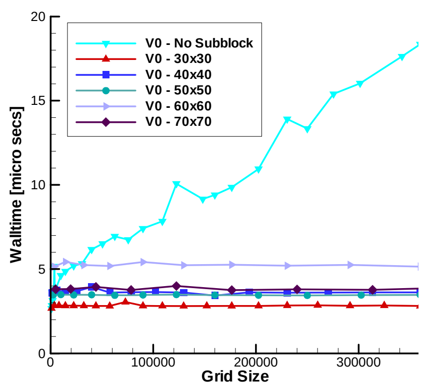


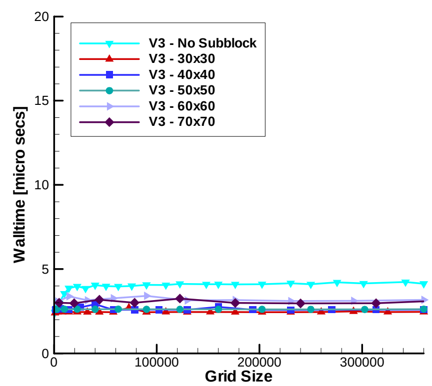




GHOST Performance

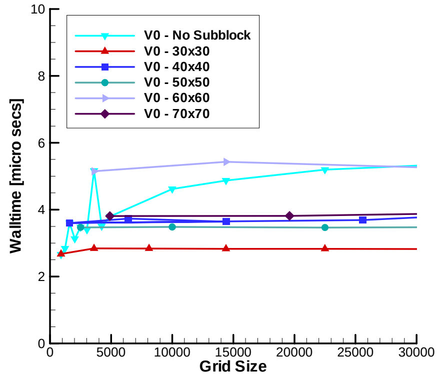
GHOST	KFC4
Laminar Steady	V0, V3

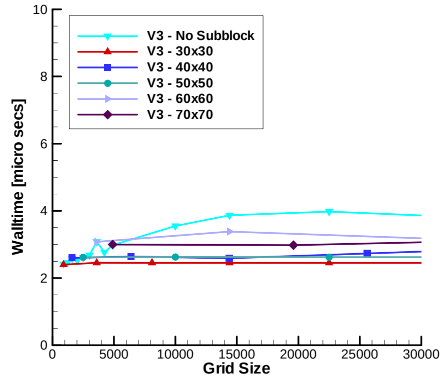




GHOST Performance

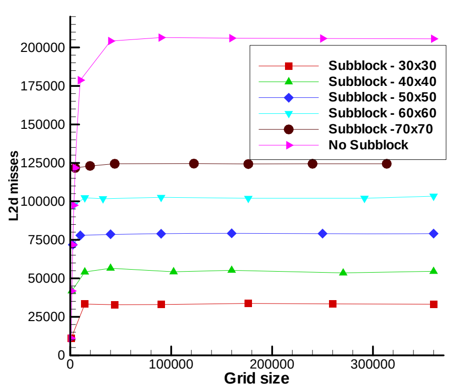
GHOST	KFC4
Laminar Steady	V0, V3

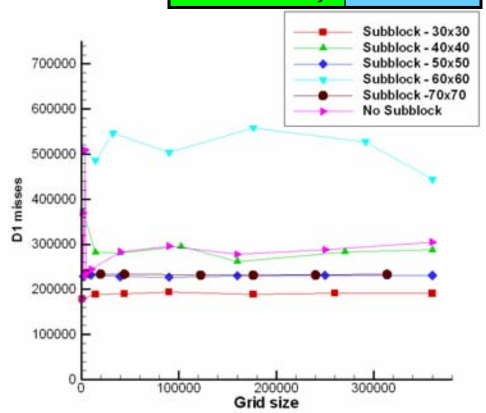




GHOST Performance

GHOST	KFC4
Laminar Steady	V3



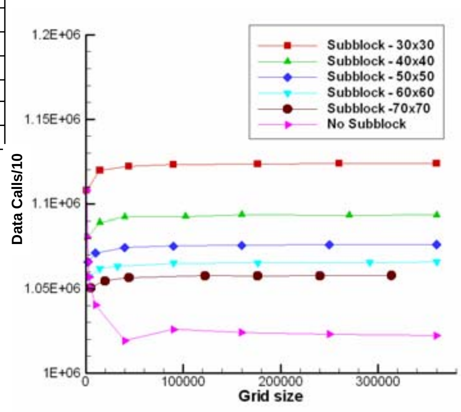


GHOST Performance

GHOST	KFC4
Laminar Steady	V3

Block Size	Walltime/Grid Point/Iteration [μ secs]		% Decrease Compared to No Block	
	V0	V3	V0	V3
No Block	18.83	4.63	-	-
70 x 70	4.55	3.48	75.8%	24.8%
60 x 60	4.75	3.40	74.7%	26.56%
50 x 50	4.12	3.33	78.1%	28.07%
40 x 40	3.97	3.31	78.9%	28.5%
30 x 30	3.69	3.29	80.4%	28.9%

Block Size	L2D Cache Miss Rate	D1 Cache Miss Rate
	V3	V3
No Block	2.0%	2.8%
70 x 70	1.2%	2.2%
60 x 60	1.0%	4.9%
50 x 50	0.7%	2.1%
40 x 40	0.5%	2.6%
30 x 30	0.3%	1.7%

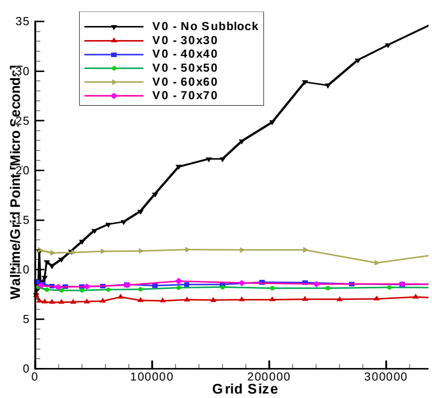


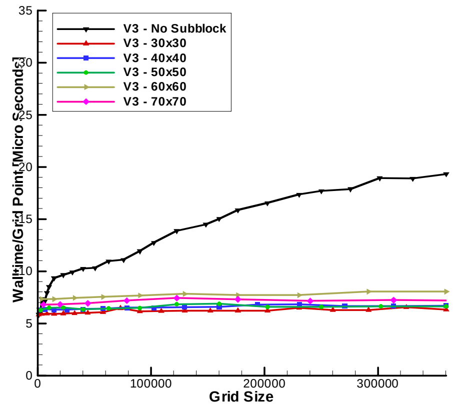
GHOST Performance - Turbulence

- Turbulence adds more equations, more routines, more variables
- Codes run with laminar optimizations only or with additional optimizations applied to turbulent routines
- Menter’s SST model

GHOST Performance - Turbulence

GHOST	KFC4
Turbulent Steady	V0t, V3t

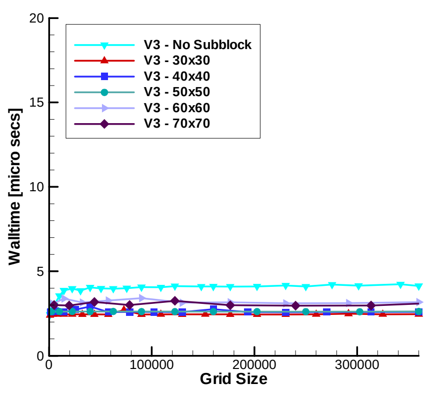


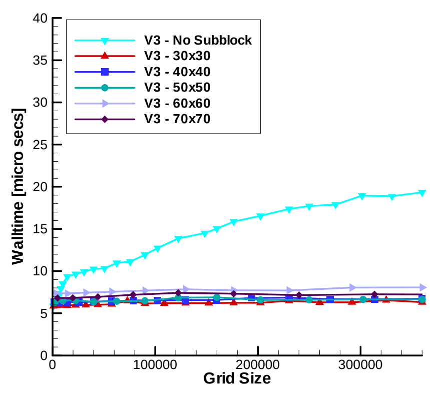


January 10, 2007 45th Aerospace Sciences Meeting and Exhibit

GHOST Performance - Turbulence

GHOST	KFC4
Lam/Trb Steady	V3, V3t



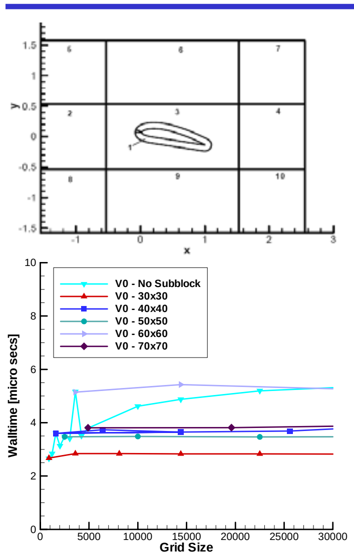


January 10, 2007 45th Aerospace Sciences Meeting and Exhibit

GHOST Performance – Unsteady

- Unsteady versions of UNCLE and GHOST use subiterations in implicit time schemes
- Unsteady subiterations reuse most steady routines
- Number of subiterations may be fixed or convergence dependent
- GHOST - use internal blocking to simplify with more complex geometries

GHOST Performance - Unsteady



GHOST	KFC4
Lam Unsteady	V0, internal

Zone	Grid Size *i x j*	Block Size *ni x nj*	Performance Improvement
1	273 x 120	30 x 30	39%
2	50 x 100	50 x 50	10%
3	300 x 100	30 x 30	32%
4	50 x 100	50 x 50	10%
5	50 x 40	50 x 40	0%
6	300 x 40	30 x 40	29%
7	50 x 40	50 x 40	0%
8	50 x 40	50 x 40	0%
9	300 x 40	30 x 40	29%
10	50 x 40	50 x 40	0%

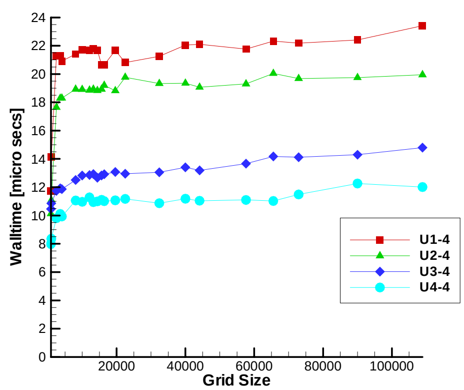
Weighted average = 31%
Actual improvement = 27%

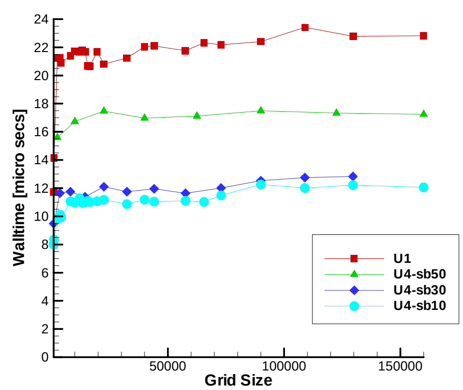
UNCLE Optimization

- Verison 1 (U1) - Cleaned original code
- Version 2 (U2) - Loop restructuring, data restructuring
- Version 3 (U3) - Loop blocking (default 100 cells)
- Version 4 (U4) - U2 + U3
- External Blocking - Blocking at grid generation (minimal gain)
- Turbulence (T1-T4) - No turbulence improvements
- Turbulence (T5) - T4 with extension of improvements to turbulence routines
- Unsteady - Incorporation of steady subiteration into unsteady code
- 3D - extension of some improvements to 3D

UNCLE Performance

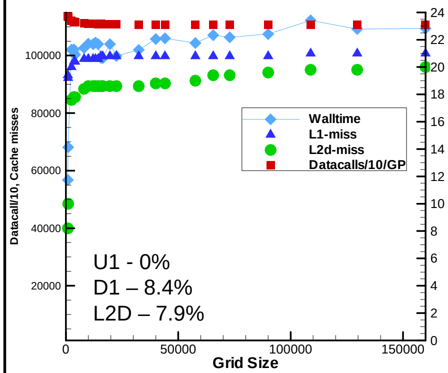
UNCLE	KFC4
Laminar Steady	U1-U4

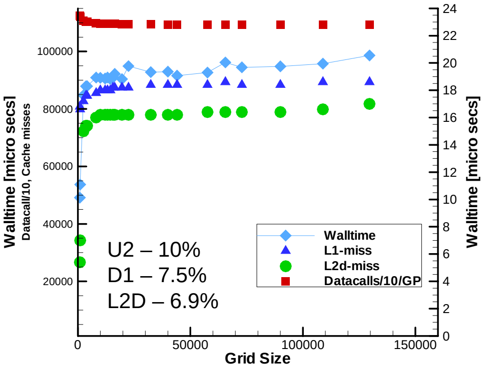




UNCLE Performance

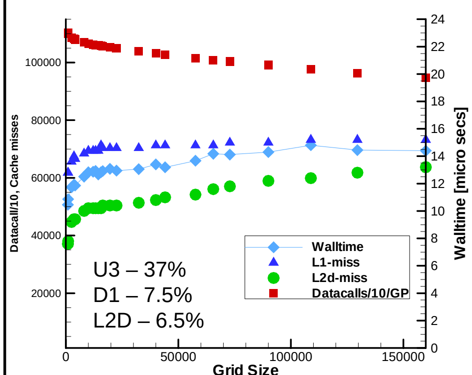
UNCLE	KFC4
Laminar Steady	U1, U2

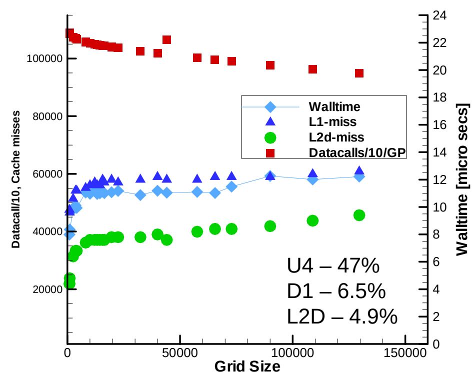




UNCLE Performance

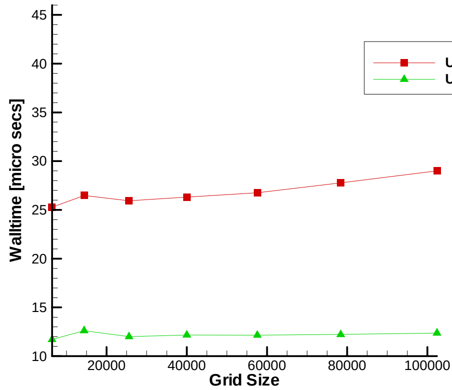
UNCLE	KFC4
Laminar Steady	U3, U4

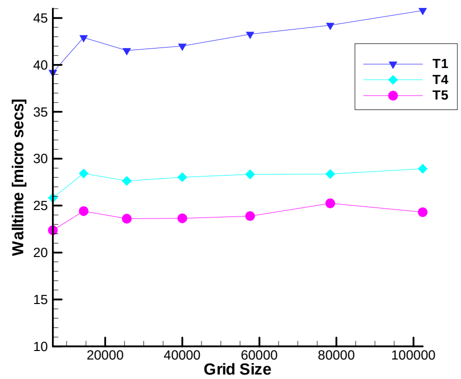




UNCLE Performance - Turbulence

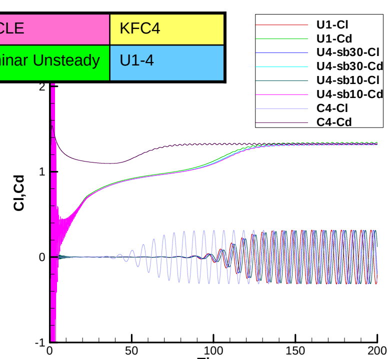
UNCLE	KFC4
Turbulent Steady	U1,4; T1,4,5

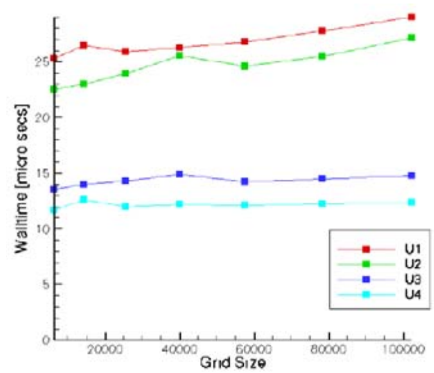




UNCLE Performance - Unsteady/3D

UNCLE	KFC4
Laminar Unsteady	U1-4

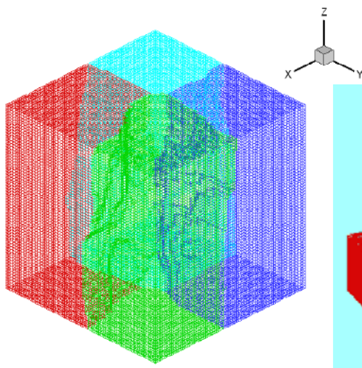




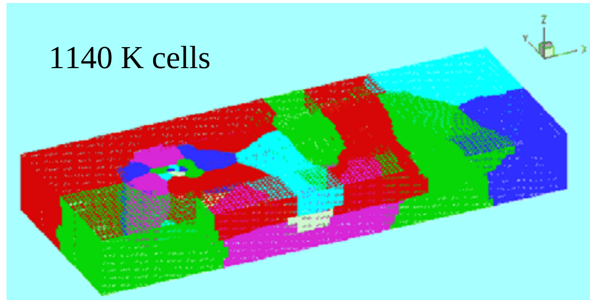
Source (Re = 100)	C_L	C_D	S_t	Speedup Comparison
Hua et al. [13]	± 0.314	1.325 ± 0.008	0.165	-5%
Ku et al. [20]	± 0.228	1.33~1.358	0.1675	-
U1	± 0.317	1.335 ± 0.008	0.1657	0
U4-sb30	± 0.3128	1.323 ± 0.008	0.1646	45%
U4-sb10	± 0.3134	1.3232 ± 0.010	0.1645	53%

UNCLE Performance - 3D

UNCLE	KFC4
Lam Un/steady	U1,3 3D



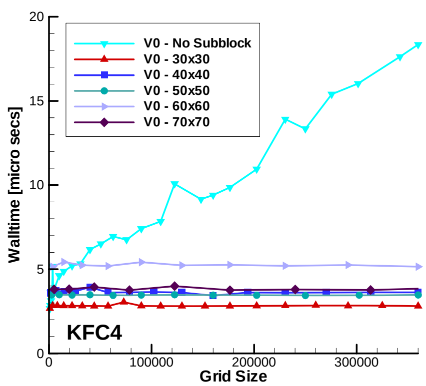
288 K cells

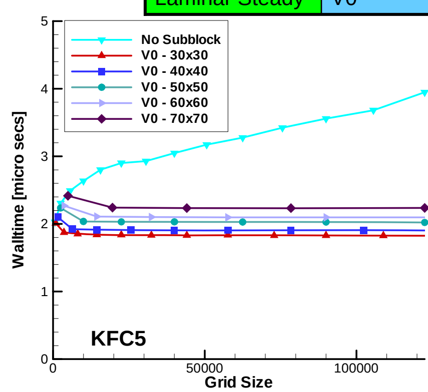


Steady - 3D laminar driven cavity, 10% reduction U1 to U3
Unsteady - 3D laminar flow over cylinder, 23% reduction U1 to U3

GHOST Performance – KFC5

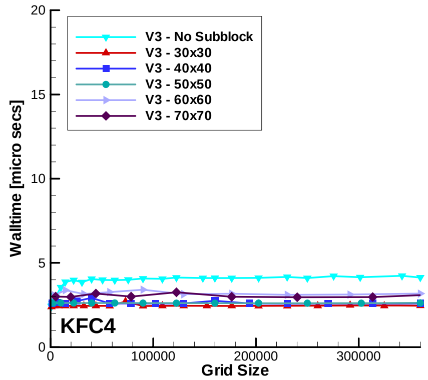
GHOST	KFC4, 5
Laminar Steady	V0

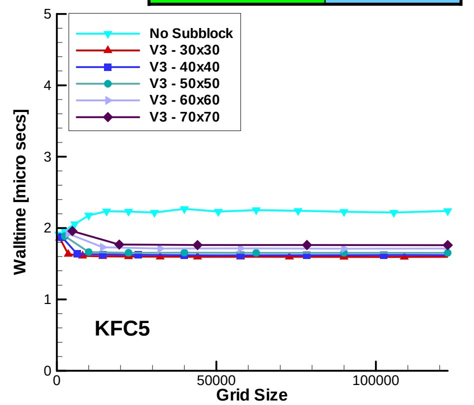




GHOST Performance – KFC5

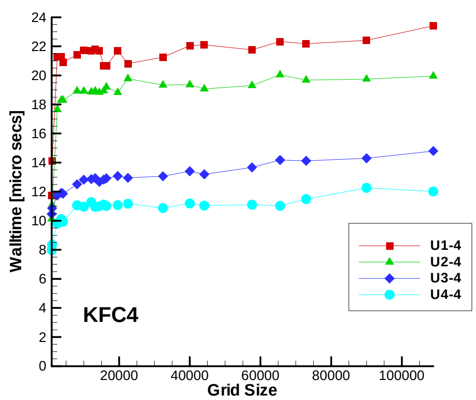
GHOST	KFC4, 5
Laminar Steady	V3

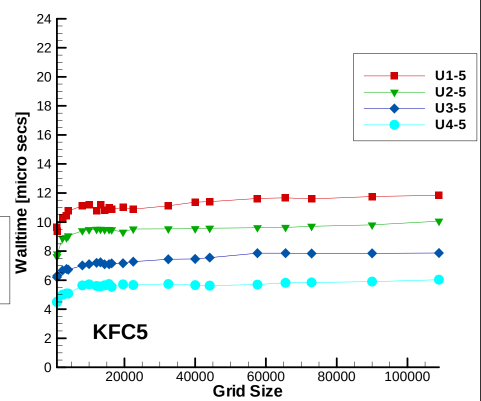




UNCLE Performance – KFC5

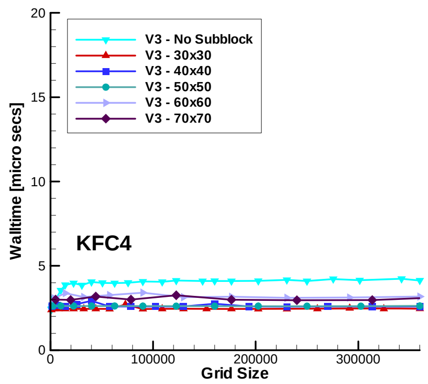
UNCLE	KFC4, 5
Laminar Steady	U1-U4

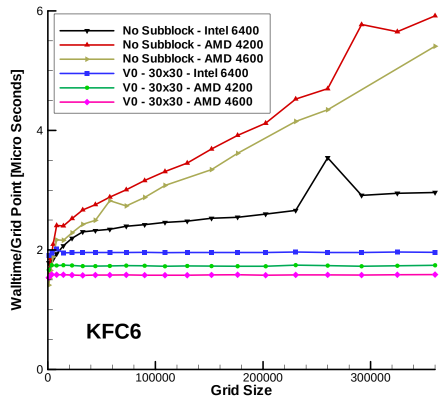




GHOST Performance – KFC6

GHOST	KFC4, 6
Laminar Steady	V0

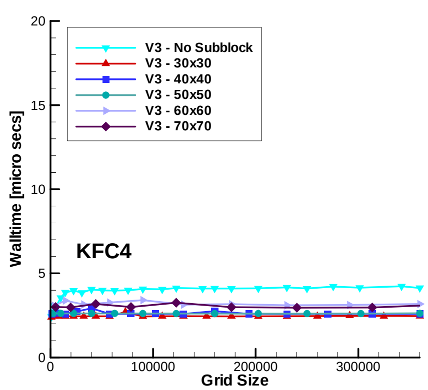


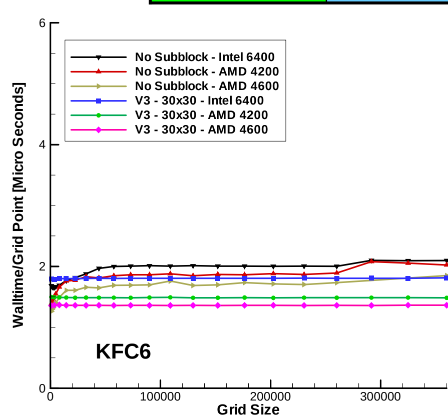


January 10, 2007 45th Aerospace Sciences Meeting and Exhibit

GHOST Performance – KFC6

GHOST	KFC4, 6
Laminar Steady	V3





January 10, 2007 45th Aerospace Sciences Meeting and Exhibit

GHOST Performance – KFC6

GHOST	KFC6
Laminar Steady	V0, V3

V3

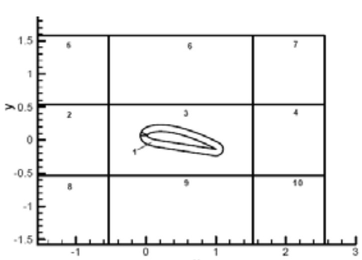
Block Size	AMD 4600	AMD 4200	Intel 6400
30 x 30	21.55 %	22.78 %	11.18 %
40 x 40	21.05 %	22.18 %	13.6 %
50 x 50	20.56 %	21.94 %	14.66 %
60 x 60	17.44 %	18.91 %	15.28 %
70 x 70	17.02 %	17.92 %	15.11 %

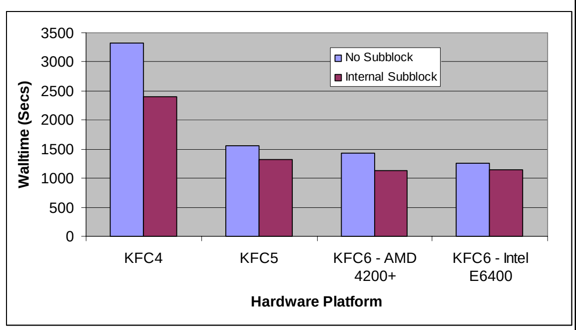
V0

Block Size	AMD 4600	AMD 4200	Intel 6400
30 x 30	60.35%	61.58%	29.55%
40 x 40	59.11%	60.64%	30.85%
50 x 50	56.20%	57.25%	30.83%
60 x 60	52.73%	54.68%	30.51%
70 x 70	52.9%	54.23%	29.43%

GHOST Performance – KFC6

GHOST	KFC4
Lam Unsteady	V0, internal





Conclusions

- Reducing L2 cache miss rates from a few percent to a few tenths of a percent can yield significant performance acceleration
- Performance enhancements on core code, simplifed flow do extend to significant gains on more complex flows, code versions
- GHOST near-cache optimized, approaching user-ready (once bugs are fixed), exportable enhancements
- UNCLE about halfway to cache optimized
- Newer commodity processors are more cache-aware, reducing the gains available through cache performance
- Reducing cache misses also reduces TLB misses