Multicore Acceleration of the Complex Ambiguity Function

D.P. Enright, E.M. Dashofy, M. AuYeung, R.S. Boughton, J.M. Clark, and R. Scrofano, Jr.

The Aerospace Corporation
2350 E. El Segundo Blvd.
El Segundo, CA 90245-4691

{Douglas.P.Enright,edashofy,mauyeung,RScott.Boughton,JMatt.Clark,Ronald.Scrofano}@aero.org

High Performance Embedded Computing (HPEC) Workshop

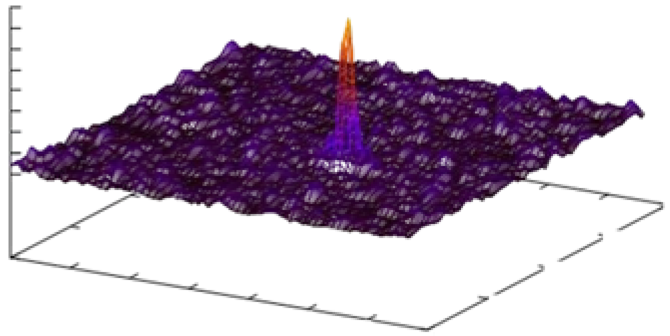
23-25 September 2008

Multicore Acceleration of the Complex Ambiguity Function

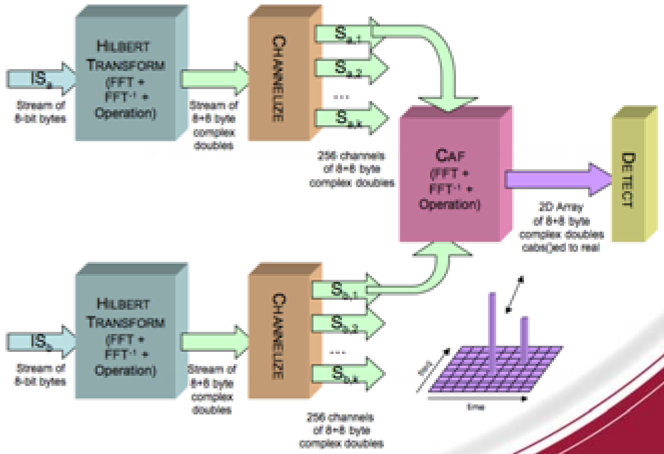
D.P. Enright, E.M. Dashofy, M. AuYeung, R.S. Boughton, J.M. Clark, and R. Scrofano, Jr.

- Performed Multicore Parallelization Study of the Complex Ambiguity Function (CAF)
- CAF is a key algorithm for performing time and frequency delay of arrival (TDOA/FDOA) for actively sensed objects
  - *Output is "caf surfaces"*
  - *Peak detection gives TDOA/FDOA*
- Entire CAF evaluation requires pre-/post-processing steps
- CAF module is most computationally intensive
- OpenMP loop-level parallelization strategy employed
- Parallel speedups of 75% or greater for dual-quad core Intel Xeon system

"caf surface"



CAF Processing Chain



IS_a
Stream of 8-bit bytes
HILBERT TRANSFORM (FFT + FFT^{-1} + Operation)
Stream of 8+8 byte complex doubles
CHANNELIZE
$S_{a,1}$
$S_{a,2}$
...
$S_{a,k}$
256 channels of 8+8 byte complex doubles
CAF (FFT + FFT^{-1} + Operation)
2D Array of 8+8 byte complex doubles cabs()ed to real
DETECT
IS_b
Stream of 8-bit bytes
HILBERT TRANSFORM (FFT + FFT^{-1} + Operation)
Stream of 8+8 byte complex doubles
CHANNELIZE
$S_{b,1}$
$S_{b,2}$
...
$S_{b,k}$
256 channels of 8+8 byte complex doubles
time

Computational Kernels & Parallelization Strategy

- Four Modules
  - *Pre-/Post-processing: HILBERTTRANSFORM, CHANNELIZE, DETECT*
    - Computational Kernels: Multiple independent FFTs, FIR filters
  - *CAF: Computationally most intensive*
    - Computational Kernel: Multiple independent cross-correlations utilizing FFTs

$$A_k(m,\nu)=\sum_{l=0}^{L-1} S_{a,k}(l+\nu)\times S_{b,k}^{*}(l)\times e^{-j2\pi ml/L}$$

caf surface $A_k(m,\nu)$

- Parallelization Strategy
  - *Large amounts of loop-level parallelism with multiple FFTs performed in parallel and no data-dependencies between loop iterations within HILBERTTRANSFORM and CHANNELIZE modules*
    - Ideal loop-structure for OpenMP `omp parallel for` iterative workshare construct
  - *CAF and DETECT modules enclosed by doubly-nested outer for-loop with multiple large FFTs performed in parallel and no data-dependencies between loop iterations*
    - Ideal loop-structure for OpenMP `omp parallel for` iterative workshare construct

Workload-Driven Evaluation

- To assess efficacy of parallelization effort and the parallel scalability of the dual-quad core system
  - *Workload was parameterized*
    - Scaling input size from 6.25 MS to 31.25 MS in steps of 6.25 MS (1MS = 2^{20} samples)
      - *Observed linear-scaling in uni-processor runtime*
    - Processing a Nominal Surface case, i.e. culled number of caf surfaces, and All Surface case, i.e. all possible caf surfaces
  - *Two Workload-Driven metrics calculated*
    - Workload-Constrained (WC) metric
      - *Ability of parallel system to minimize overall wall-clock time*

$$\mathrm{Speedup}_{\mathrm{WC}}(p \text{ cores}) = \frac{\mathrm{Time}(1 \text{ core})}{\mathrm{Time}(p \text{ cores})}$$

    - Modified Linear-Scaled Workload Time-Constrained (MLSWTC) metric
      - *Ability of parallel system to maintain a uni-core runtime as workload is increased in proportion to number of parallel resouces (cores)*

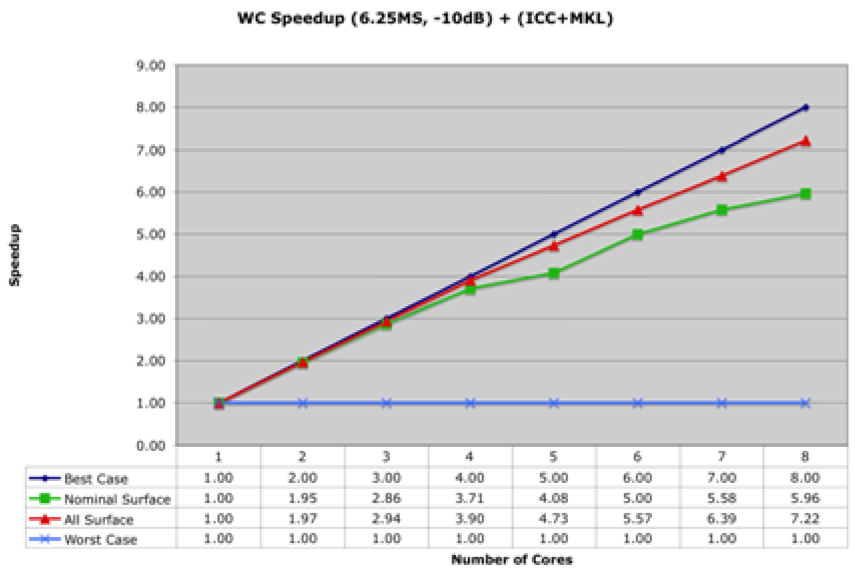
$$\mathrm{Speedup}_{\mathrm{MLSWTC}}\left(p \text{ cores, } \min(p, ubism) \text{ input size}\right) = \frac{\mathrm{Time}(1 \text{ core, } 1 \text{ input size})}{\mathrm{Time}\left(p \text{ cores, } \min(p, ubism) \text{ input size}\right)}$$

        - *ubism is the upper-bound input size multiple of base input size (5 for study)*
      - *Modification of "scaled-speedup" model of Gustafson[1]*

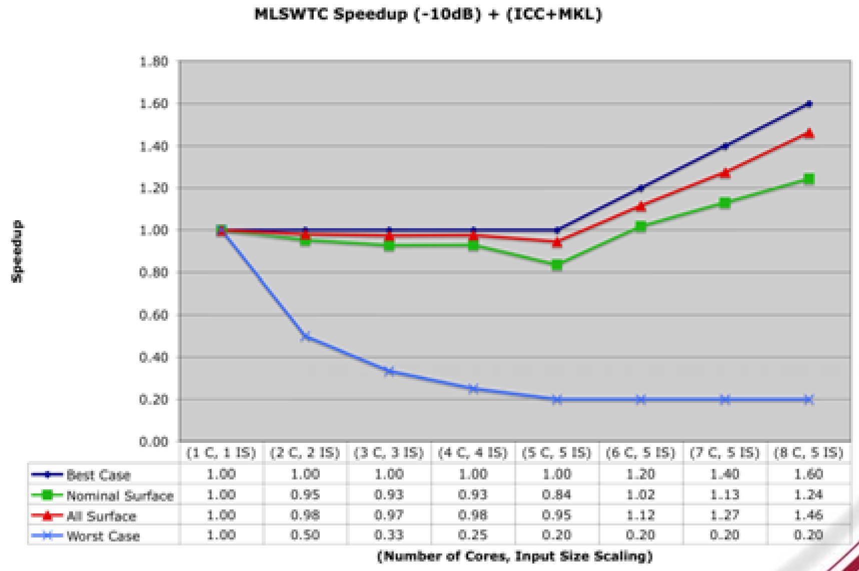
[1] *Gustafson, J.L., "Reevaluating Amdahl's Law",* Comm. ACM, *v. 31, no. 5, p. 532-533 (1988)*

Results

- Using dual-quad core Intel Xeon E5335 “Clovertown” system[1]
  - *ICC v. 10.0 + MKL v. 10.0.3.20*
- WC Speedup
  - *(6.25MS, -10dB) Workload*
    - (-10dB = 8K element CAF FFTs)
  - *Nominal Surface, 5.96/8 = 75%*
  - *All Surface, 7.22/8 = 90%*
  - *Process given workload 6x to 7.2x faster*
- MLSWTC Speedup (-10dB)
  - *(5 cores, 5 input size)*
    - Nominal Surface, .84/1
    - All Surface, .95/1
  - *Both Surface Processing Cases*
    - Process 5x base input with 6 cores in same amount of time as uni-core processor base input runtime



	1	2	3	4	5	6	7	8
Best Case	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00
Nominal Surface	1.00	1.95	2.86	3.71	4.08	5.00	5.58	5.96
All Surface	1.00	1.97	2.94	3.90	4.73	5.57	6.39	7.22
Worst Case	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00



	(1 C, 1 IS)	(2 C, 2 IS)	(3 C, 3 IS)	(4 C, 4 IS)	(5 C, 5 IS)	(6 C, 5 IS)	(7 C, 5 IS)	(8 C, 5 IS)
Best Case	1.00	1.00	1.00	1.00	1.00	1.20	1.40	1.60
Nominal Surface	1.00	0.95	0.93	0.93	0.84	1.02	1.13	1.24
All Surface	1.00	0.98	0.97	0.98	0.95	1.12	1.27	1.46
Worst Case	1.00	0.50	0.33	0.25	0.20	0.20	0.20	0.20

[1]2.0GHz; 2x4MB shared L2, 32KB I/D L1; 10.6 GB/sec FSB; 8GB system memory; 64-bit CentOS5