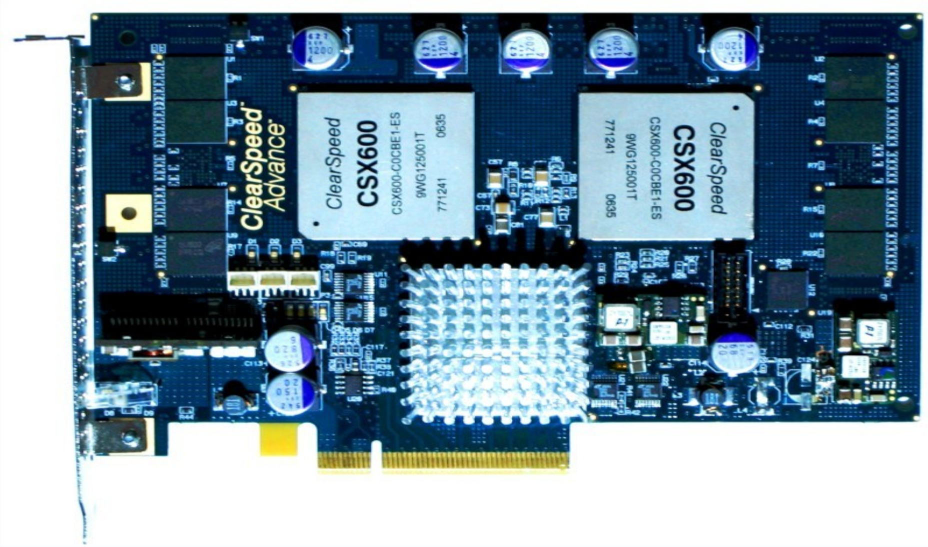
Clearspeed Acceleration Applied to Passive Coherent Location

Joe Milburn and *Khomotso Kganyago

kkganyago@csir.co.za

http://www.chpc.co.za

RRSG - UCT
ACE – CHPC
*CHPC



Outline

- Passive Coherent Location with Doppler and Bearing
- Brief explanation of programming model
- Acceleration through parallelization and host side libraries
- Benchmarking on eight Clearspeed accelerated nodes
- Conclusions

Where are we?

The first phase of computer procurement and integration was completed in May 2007, with the official launch on the 22nd of May 2007, and the operations officially starting on the 1st of June 2007. The first phase computer consists of an IBM e1350 Linux computer cluster with 160 nodes:

- Each node is equipped with two dual-core AMD Opteron 2.6GHz processors and 16GB of random access memory, giving an aggregate of 640 processing elements with a capacity of approximately 2.5 Tflops.
- Eight nodes are equipped with Clearspeed Advance e620 Accelerator boards each with two CSX600 processors with 96 processing elements running at 210 megahertz.
- Nodes are interconnected by an Infiniband network at 10Gbps
- In addition to local hard disks, all nodes have access to a shared storage system with a capacity of 54TB using General Parallel File System (GPFS)
- Runs with SLES 10 Linux operating systems

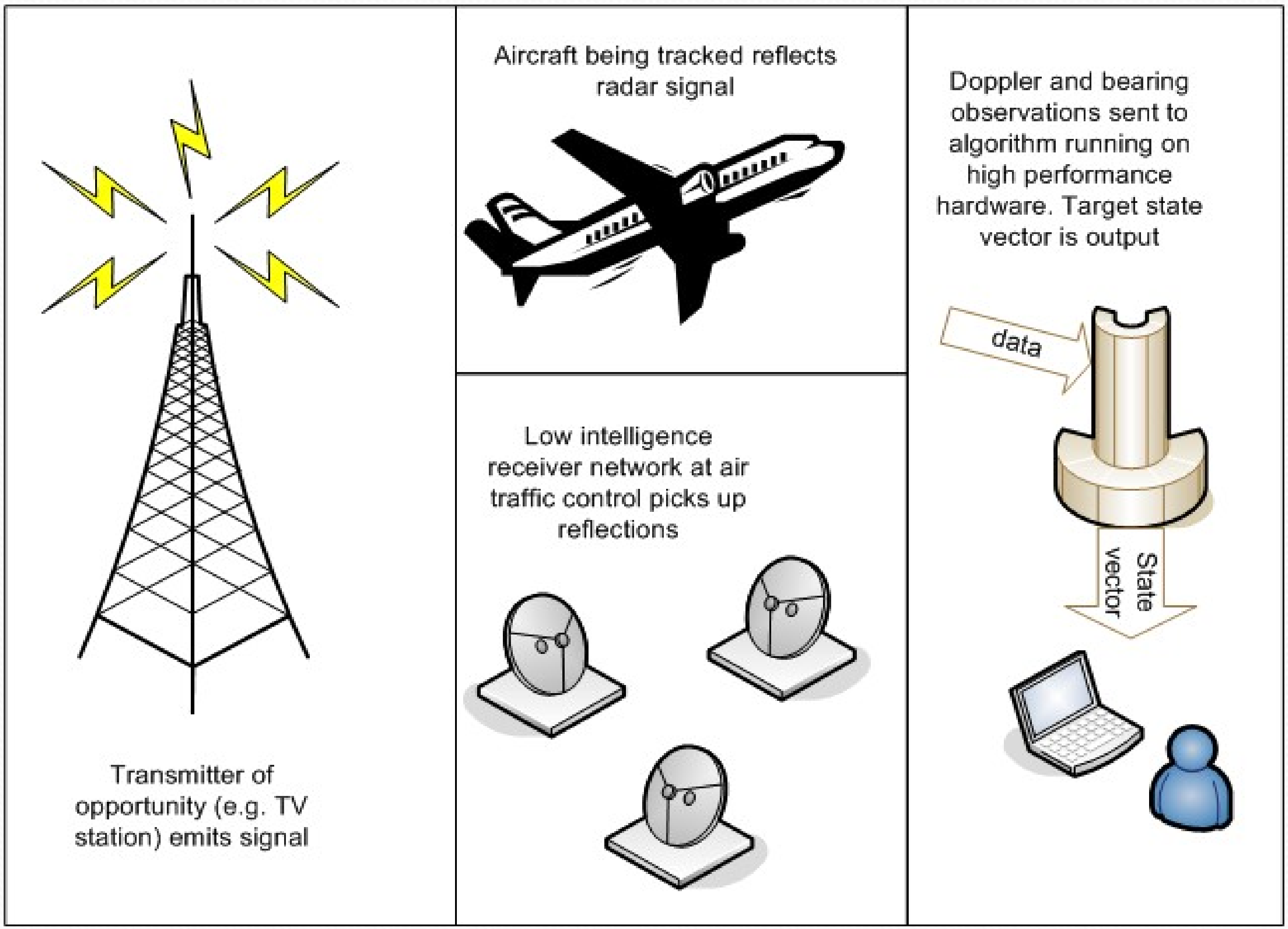
- Two P690 SMP machines:
- 32 x 1.9GHz Power4+ CPUs and at least 32GB of RAM each

Background

- Africa has one of the largest aircraft accident rates per flying hour in the world.
- One factor that contributes to unsafe flying conditions is the lack of infrastructure and equipment required for Aircraft Traffic Control (ATC)
- PCL radar uses a network of receivers to track targets through their backscatter from existing Continuous Wave (CW) transmissions, such as broadcast TV or radio.
- The lack of an active transmitter in a PCL system results in relatively low procurement, operation and maintenance costs


pictures courtesy of www.aloha.net and www.nytimes.com

Passive Coherent Location with Doppler and Bearing



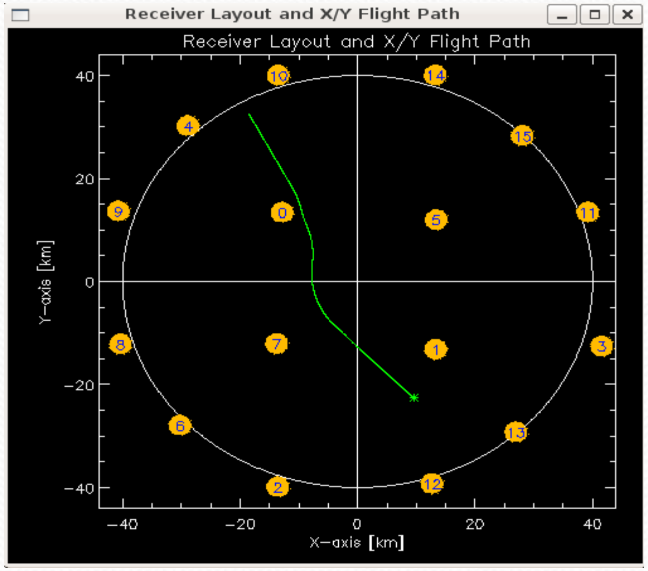
$$\begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix}$$

state vector

Gauss Newton Algorithm

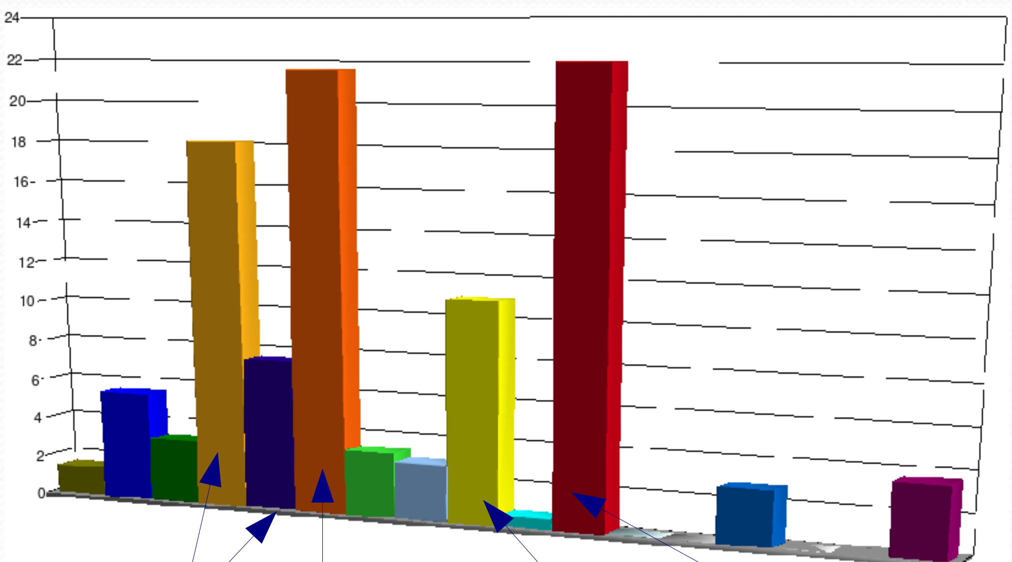
- PCL tracking algorithm under consideration developed by Dr Norman Morrison, Dr Richard Lord and Professor Michael Inggs of UCT
- Part of ongoing investigation into PCL radar systems jointly undertaken by the University of Cape Town and the University College of London
- Algorithm takes advantage of the fact that as computer processing becomes cheaper and faster, expensive hardware can be replaced by mathematical computation
- Algorithm processes Doppler and bearing data to produce a state vector estimation of the target. This specifies position, velocity and acceleration of the target in a three dimensional Cartesian coordinate system

- System uses non-linear differential correction, commonly known as the Gauss-Newton method. It involves intensive double precision floating point computation and high dimension matrix multiplication and inversions



Application and Algorithm

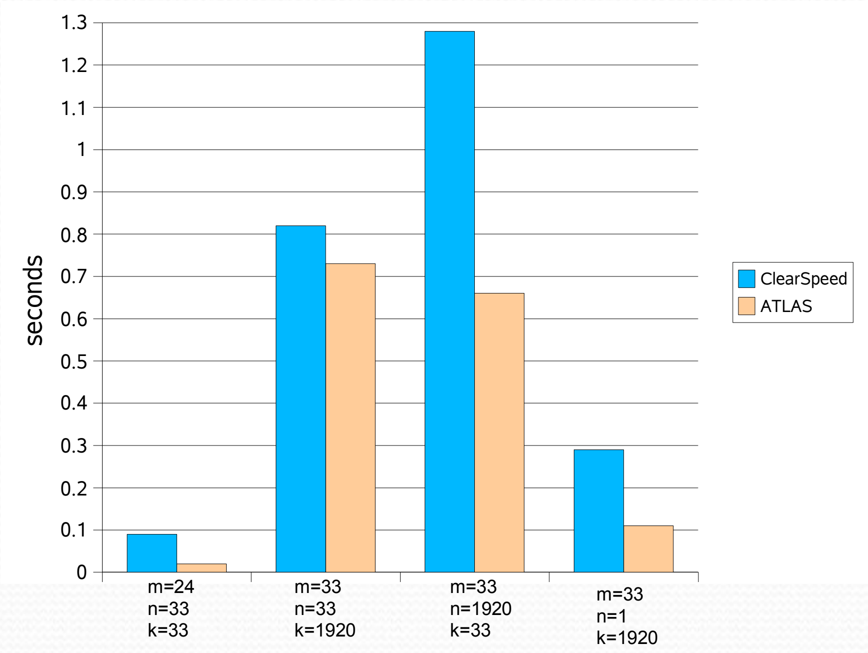
- The PCL tracking algorithm under consideration uses non-linear differential correction, commonly known as the Gauss-Newton method. It involves intensive double precision floating point computation and high dimension matrix multiplication and inversions. The algorithm profile is shown below left.
- Intended method of acceleration:
  - (1) Use of accelerated CSXL dgemm routines.
  - (2) Parallelization through loop unrolling.

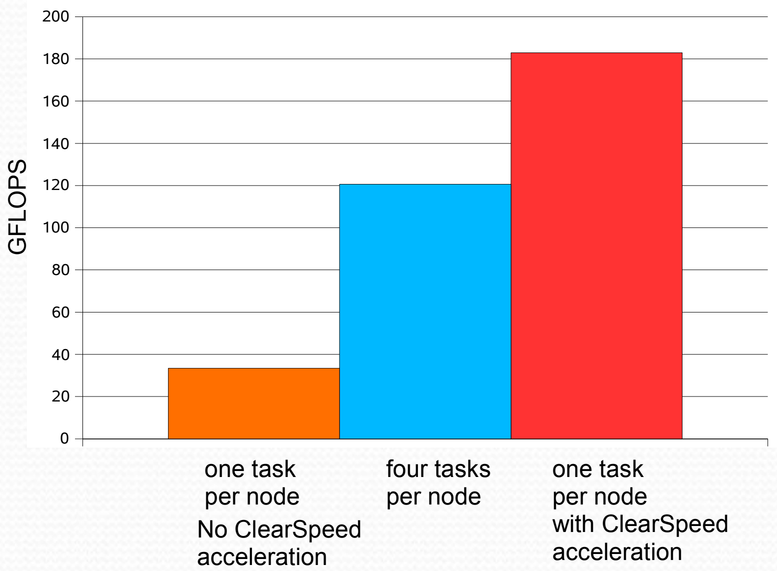


- System specification:
  - 4x Intel(R) Xeon(TM) dual-core CPUs
  - 2GB RAM, SLES 9
  - ClearSpeed Advance X620
  - Software package version 2.50

Results so Far

- Testing performance of CSXL's dgemm vs the host's ATLAS BLAS library version determined that the matrix sizes used in the algorithm are unfortunately not candidates for ClearSpeed acceleration (below left).
- Preliminary results of parallelization through loop unrolling is encouraging, however further optimization (through vectorization etc.) and rearranging of the code to allow the host and ClearSpeed card to operate concurrently are required before serious performance benefits will be seen.
- Results of benchmarking on eight ClearSpeed accelerated nodes on the CHPC cluster are also shown below right





Work in progress

- The Physics group at UCT and iTemba Labs compiled and linked, with the assistance of Nick Allsopp (IBM), the ROOT Framework <http://root.cern.ch/root/> used for data analysis.
- A 10% increase in performance as a result was reported, still to be verified by the project users.

Work in progress

Key	Title	SIG(s)	Software Requirements	Proprietary (P), Open Source (O) or	Parallel (P) or Serial	Number of		Code Descriptor
						Minimum	Maximum	
1	Large Scale Simulations of Energy Storage Materials	Physics	DL_POLY [*http://www.cse.scitech.ac.uk/ccg/software/DL_POLY/*]	P*	P	1	1000	Molecular Dynamics Simulation
2	Computational Space and Astrophysics	Physics Space & Astrophysics	Fortran 90 [*http://gcc.gnu.org/wiki/Gfortran*] MPI	O O	S P	1 1	1 ~	Fortran Compiler Message Passing Library
3	Ocean Atmosphere Modelling	Oceangraphy	ROMS [*http://www.myroms.org/*] WRF [*http://www.wrf-model.org/index.php*] Fortran 90 [*http://gcc.gnu.org/wiki/Gfortran*]	O O O	P P S	1 1 1	96 128 1	Ocean Model Climate Model Fortran Compiler
3	Multi-model probabilistic seasonal climate projections	Climatology	WRF [*http://www.wrf-model.org/index.php*] NCAR CAM [*http://www.ccsm.ucar.edu/models/atm-cam/*] The Unified Model System [*http://tinyurl.com/3muw36*] Fortran 90 [*http://gcc.gnu.org/wiki/Gfortran*]	O O O O	P P P S	1 1 1 1	128 30 ~ 1	Weather Model Atmosphere Model Ocean & Atmosphere Model Fortran Compiler
4	Monte Carlo simulations of	Physics	BLAS [*http://www.netlib.org/blas/*] LAPACK [*http://www.netlib.org/lapack/*]	O O	S S	1 1	1 1	Basic Linear Algebra Routines
5	A South African High Performance Multi-physics Computational Fluid Dynamics Solver	Computational Mechanics and Electromagnetics	GCC Compilers [*http://gcc.gnu.org/*] GAMBIT mesh generator [*http://tinyurl.com/3kb8ly*]	O P	S S	1 1	1 1	C++ Compiler Mesh Generator
6	Grand Challenge Electromagnetic Computer Simulation for the MeerKAT and SKA	Computational Engineering	FEKO [*http://www.feko.info/feko-product-info*] Fortran 95 [*http://gcc.gnu.org/wiki/GFortran*] LAPACK [*http://www.netlib.org/lapack/*] BLAS [*http://www.netlib.org/blas/*] MPI	P O O O O	P S S S P	1 1 1 1 1	64 1 1 1 ~	Electromagnetic Fortran Compiler Linear Equation Solving Routines Basic Linear Algebra Routines Message Passing Library
7	Modelling HIV-1 evolution	Bioinformatics	HyPhy [*http://www.hyphy.org/*] PAML [*http://abacus.gene.ucl.ac.uk/software/paml.html*] R [*http://cbio.uct.ac.za/CRAN/*] GCC compilers [*http://gcc.gnu.org/*] GSL - GNU Scientific Library [*http://www.gnu.org/software/gsl/*] PhyML [*http://atgc.lirmm.fr/phyml/*] BEAST [*http://beast.bio.ed.ac.uk/*]	O P* O O O P* O	P S S S S S S	1 1 1 1 1 1 1	~ 1 1 1 1 1 1	Genetic Sequence Analysis Phylogenetic Analyses of DNA Statistical Computing and Graphics Compiler Suite Numerical Library Algorithm to Estimate Large Phylogenies Bayesian Analysis of Molecular Sequences
8	Nuclear Collisions and Data Grid for the Physics Community	Physics Grid	Scientific Linux [*https://www.scientificlinux.org/*] GCC compilers [*http://gcc.gnu.org/*] GNU make [*http://www.gnu.org/software/make/*] Autotools [*http://www.gnu.org/software/autoconf/*] gLite [*http://glite.web.cern.ch/glite/*] ROOT [*http://root.cern.ch/*]	O O O O O O	S S S S P S	1 1 1 1 2 1	1 1 1 1 ~ 1	Linux Distribution Compiler Suite Automated Compilation Program Portability Tools Grid Computing Middleware Data Analysis Framework
9	Modern South African Astronomy and Cosmology: Comfronting the Simulated and the Observed Universe	Astronomy	GADGET-2 [*http://www.mpa-garching.mpg.de/gadget/*] GCC compilers [*http://gcc.gnu.org/*] Fortran 90 [*http://gcc.gnu.org/wiki/Gfortran*] Python [*http://www.python.org/*] NumPy [*http://numpy.scipy.org/*] SciPy [*http://www.scipy.org/*] MPI OpenMP [*http://www.openmp.org/*] IDL [*http://www.ittvis.com/idl/*] NAG [*http://www.nag.com/*] MATLAB [*http://www.mathworks.com/products/matlab/*]	O O O O O O O O P P P	P S S S S S P P S S/P S/P	1 1 1 1 1 1 1 1 1 1 1	~ 1 1 1 1 1 ~ ~ 1 ~ ~	Cosmological Simulations Compiler Suite Fortran Compiler Scripting Language Numerical Routines for Python Scientific Routines for Python Message Passing Library Shard-Memory Parallel Programming Data Visualization & Analysis Numerical Algorithms Library Interactive Programming Environment

Conclusions

- The Clearspeed programming model has a less than two week learning curve, as it is fairly straight forward. Users experienced in parallel programming will find ClearSpeed programming even easier to adjust to.
- Testing shows that the matrix sizes of the Gauss-Newton tracking algorithm are not good candidates for Clearspeed acceleration
- Unrolling of loops could possibly result in acceleration, but only if the algorithm is rewritten to allow overlapping of computation and data transfer, as not enough of the algorithm is data parallel to see significant performance benefits otherwise. The feasibility of doing this is still being explored.
- Recommendation: Esoteric “poly memory exceeded” errors should at least be accompanied by a more readable explanation.
- Due to the hefty price tag of the card, significant performance gains are required to make porting applications feasible.