



Searching Multidimensional Parameter Spaces to Optimize Parallel I/O

Kate Ericson
SIParCS 2009
John Dennis (NCAR)
Michelle Strout (CSU)

August 5, 2009

Outline

Overview

- Configurations

- Machines

Software

- Search Methods

 - Active Harmony

 - Stochastic Local Search

- Approach Comparison

Results

- Frost Results

- Kraken Results

- Observations of Methods

Conclusions and Future Work

- Conclusions

- Future Work

Outline

Overview

Configurations

Machines

Software

Search Methods

Active Harmony

Stochastic Local Search

Approach Comparison

Results

Frost Results

Kraken Results

Observations of Methods

Conclusions and Future Work

Conclusions

Future Work

Overview

- We are looking at introducing parallel I/O into Community Climate System Model (CCSM)
 - Using the Parallel I/O (PIO) library
 - MPI-IO based I/O library in CCSM4
 - Currently looking at testpio which simulates I/O in Parallel Ocean Program (POP)
- Questions about the parameter space:
 - Are there generic 'good' PIO configurations?
 - What characterizes a 'good' PIO configuration?
 - Can we find something better than an educated guess?
- How to explore the parameter space?
 - Active Harmony
 - Stochastic Local Search



Note on Configurations

- PIO configurations are not Computational configurations
 - Some traits may be shared, such as number of cores
 - One Computational configuration has many possible PIO configurations

Computer Configuration



PIO Configuration





Example PIO Configuration

- Two parameters discussed later
 - I/O Format:
 - snc - Serial NetCDF
 - pnc - Parallel NetCDF
 - Rearrangement:
 - none
 - box

Computer Configuration



PIO Configuration





Meet the Machines

- Frost
 - Blue Gene/L at NCAR
 - Relatively stable I/O
 - I/O times do not vary too much
 - 6 PIO parameters
 - 4,000,000+ configurations





Meet the Machines

- Kraken
 - Cray XT5 at NICS
 - More variability in I/O performance
 - Performance can vary by 18x
 - Discussed by Nick Jones
 - Lustre system introduces 2 new PIO parameters
 - 260,000,000,000+ configurations



Outline

Overview

Configurations

Machines

Software

Search Methods

Active Harmony

Stochastic Local Search

Approach Comparison

Results

Frost Results

Kraken Results

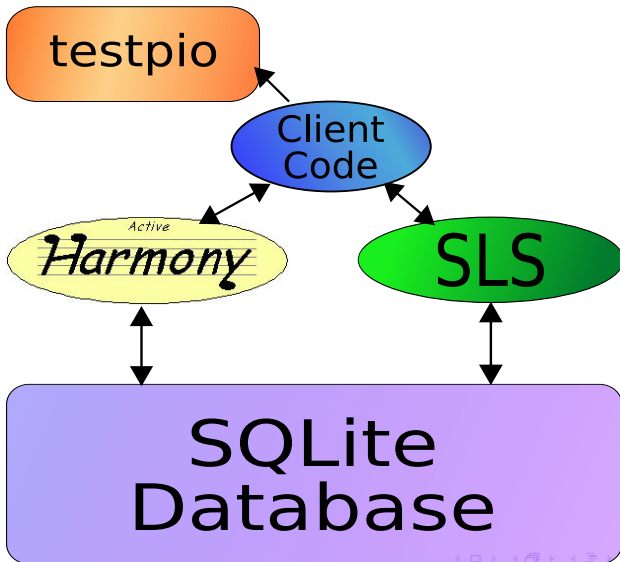
Observations of Methods

Conclusions and Future Work

Conclusions

Future Work

Software Infrastructure





- In development for the past 10 years at the University of Maryland
- Allows for different methods of search:
 - Brute Force
 - Modified Nelder-Mead

Nelder-Mead

A nonlinear implementation of the simplex method.

Stochastic Local Search

- Randomness can help searches
 - Random Restarts
 - Random Selection of Neighbors
- Currently supports
 - Simulated Annealing

Simulated Annealing

Named after metallurgical annealing – a metal is heated to excite atoms, as it is cooled the atoms settle into a new configuration, helping to remove impurities.

Approach Comparison

	Active Harmony	Simulated Annealing
pros	<ul style="list-style-type: none"> • Robust framework in place • Can specify starting position • Generates graph depicting progress 	<ul style="list-style-type: none"> • Stochastic • No dependencies • Random restarts in a run • Requires no domain knowledge
cons	<ul style="list-style-type: none"> • Deterministic • Requires some domain knowledge • Can get stuck easily - no random restart • Search implementation is difficult to modify 	<ul style="list-style-type: none"> • No guarantee of convergence • Cannot specify starting position

Outline

Overview

Configurations

Machines

Software

Search Methods

Active Harmony

Stochastic Local Search

Approach Comparison

Results

Frost Results

Kraken Results

Observations of Methods

Conclusions and Future Work

Conclusions

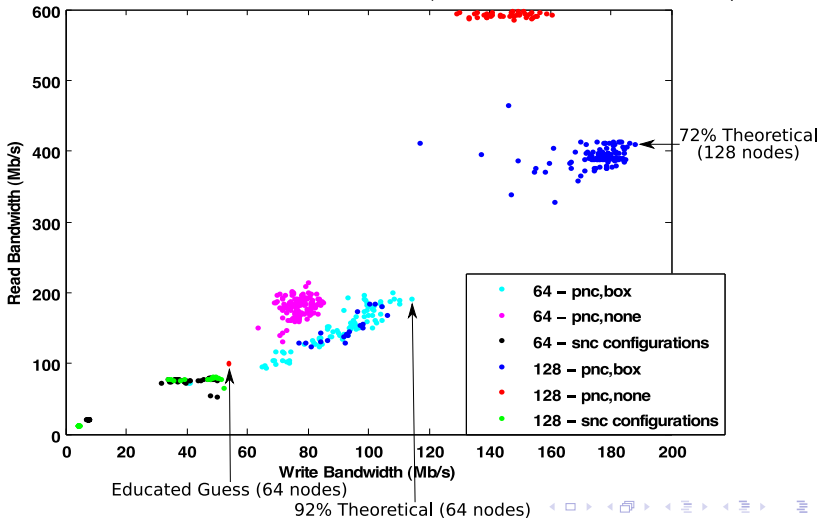
Future Work



Frost Results

- Initial tests on Frost are promising

Correlation between write and read bandwidths on Frost (64 and 128 nodes – 128 and 256 cores)

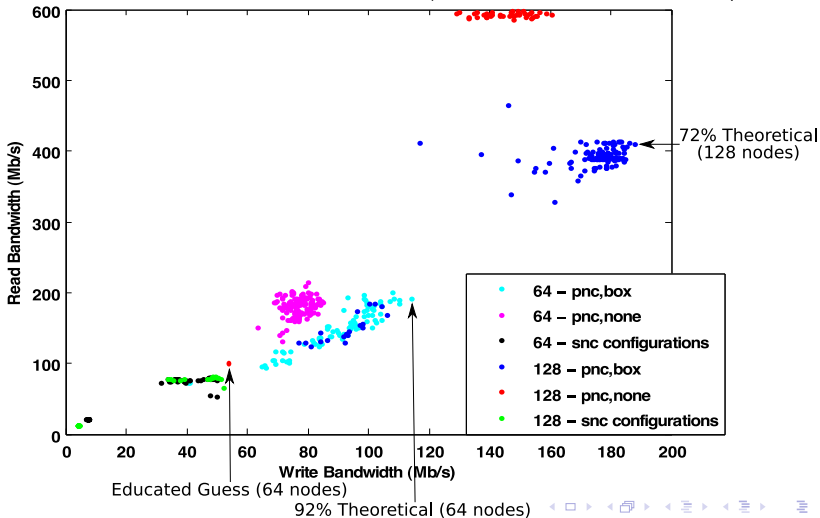




Frost Results

- Looks like some generalizations can be made

Correlation between write and read bandwidths on Frost (64 and 128 nodes – 128 and 256 cores)

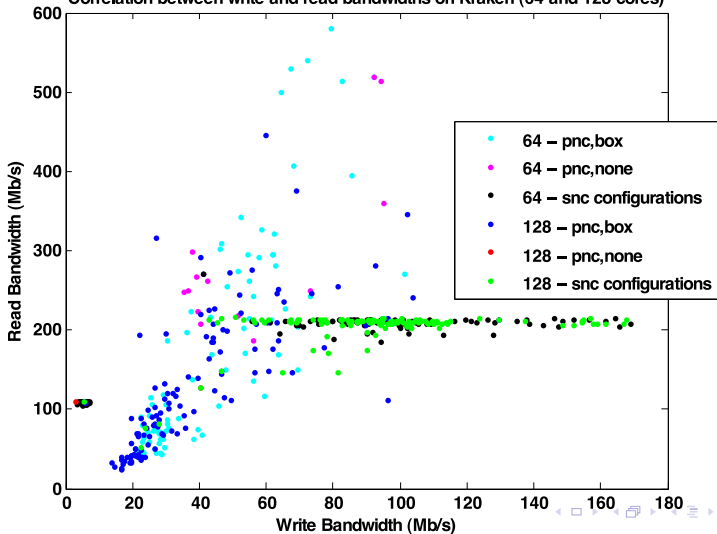




Kraken Results

- Seeing much better overall performance with 64 cores

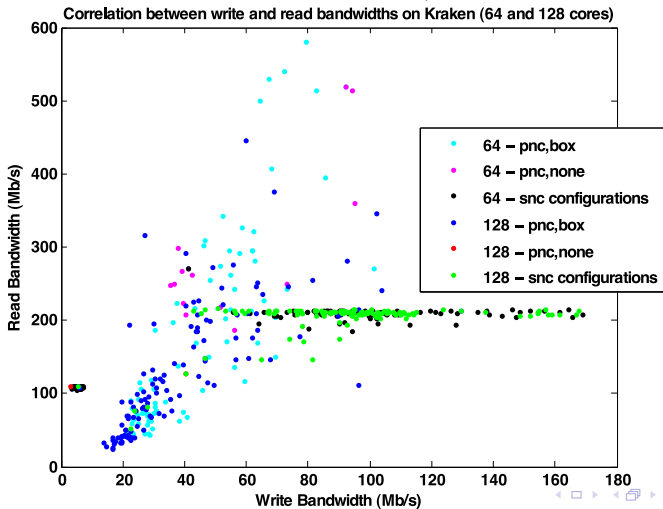
Correlation between write and read bandwidths on Kraken (64 and 128 cores)





Kraken Results

- snc configurations get generally better write speeds
- seem to be maxing out a single I/O node



Observations of Methods

- Both approaches have been promising
 - Database improves both
- Active Harmony
 - More systematic
 - More control
 - Inclined to get stuck
- Simulated Annealing
 - Current implementation depends heavily on starting position
 - Random exploration

Outline

Overview

Configurations

Machines

Software

Search Methods

Active Harmony

Stochastic Local Search

Approach Comparison

Results

Frost Results

Kraken Results

Observations of Methods

Conclusions and Future Work

Conclusions

Future Work

Conclusions

- Active Harmony and SLS work well together
 - Both search the parameter space in a different way
 - Information from one can help the other
- On Frost, we can see definite trends
 - Easy to see 'good' configurations
- Kraken is less definitive
- Serial NetCDF is more stable
 - May be preferable for machines with variable I/O performance
- We can do better than educated guesses

Next Steps

- Experiments on Kraken have been inconclusive
 - Average write speeds across multiple runs?
- Harmony has problems on Kraken
 - On slow days, Active Harmony cannot start
- Run larger experiments
 - No larger than 128 node runs
 - Only looking at 2 dimensional array
- Modify SLS
 - Different search methods
 - Make SA implementation 'smarter'



Questions?

Kate Ericson

ericson@cs.colostate.edu