tomoRecon: High-speed tomography reconstruction on workstations using multi-threading

Mark Rivers

GeoSoilEnviroCARS, Advanced Photon Source

University of Chicago

Motivation: Faster Detectors

- Time to collect tomography datasets has decreased rapidly
- New fast CMOS detectors, such as the PCO Dimax
  - 2016 x 2016 resolution
  - 1279 frames/s at full resolution
  - 4502 frames/s at half resolution (1008 x 1000)
  - Full [2016, 2016, 1200] dataset in 1 second with pink or white beam!
- This (or similar) detectors now in use at Swiss Light Source, APS, ESRF, Spring-8
- These speeds are into on-board camera memory
  - It does take several minutes to read the camera memory into the computer
- Clearly a need for high speed tomography reconstruction to keep up with data collection rates.

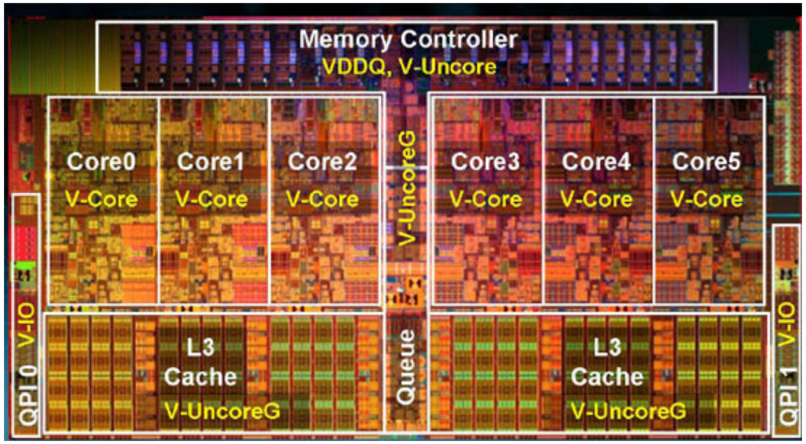
pco.dimax camera

Motivation: Goals for Reconstruction at Tomography Facilities

- Ability to reconstruct faster than data collection
  - View results from previous sample while next one is still collecting
  - Rapid feedback on data collection parameters
  - Feedback to guide the course of the study
  - Prevents backlog of data to be processed at the end of a run
- Ability to reconstruct on "affordable" computers
  - Most synchrotron beamlines reconstruct on site, users take reconstructed data home
  - Desirable for them to be able to re-do reconstruction at their home institutions
    - Change the rotation center
    - Change ring artifact removal,
    - Missing or lost files
  - Should run on Windows, Linux, or Mac

Motivation: Faster Computers

- Computers have changed remarkably in last 2-3 years
- Memory
  - Now very inexpensive
  - I recently upgraded my PC from 48 GB to 96 GB for $500!
  - So memory is $10/GB; 96 GB is less than $1,000
- Multiple CPU cores
  - CPU speeds are not increasing very much
  - Rather manufacturers are adding multiple cores to each CPU chip
  - Systems with 8 or 12 cores are now less than $3,000
- Most tomography reconstruction software does not exploit these advances



Intel six-core processor

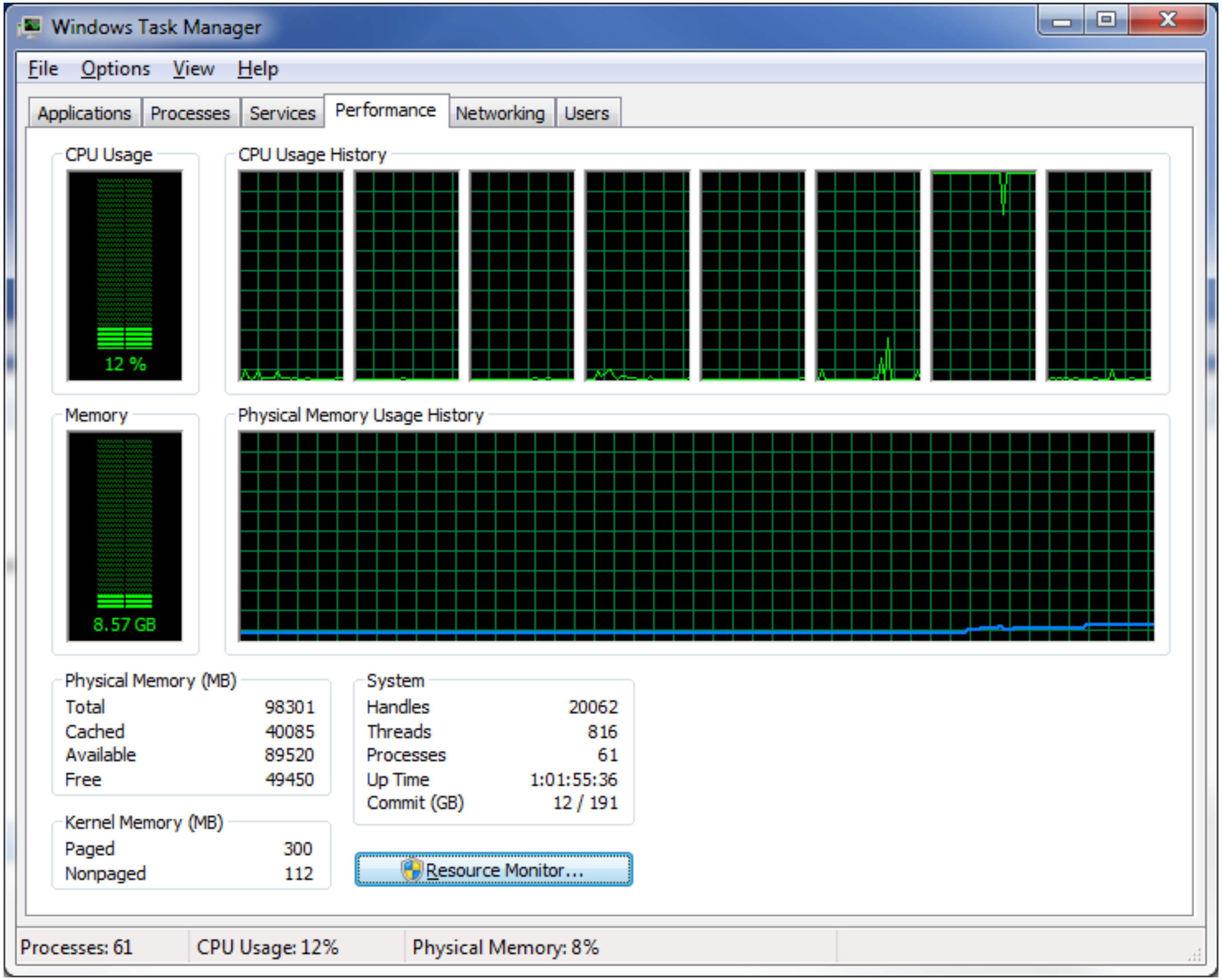
Reconstruction Times at Some Existing Tomography Facilities

Beamline	Computer	Cost	Software	Medium dataset		Large dataset	
		(~)		**Size**	**Time (s)**	**Size**	**Time (s)**
APS 2-BM	8-node Linux cluster	$70,000	Gridrec	[1392, 1040, 900]	**441**	[2048, 2048, 1500]	**2642**
Swiss Light Source TOMCAT	5-node Linux cluster	$50,000	Gridrec	[1024, 1024, 1001]	1 job: **1674** 20 jobs: **119**	[2048, 2048, 1501]	1 job: **7651** 20 jobs: **473**
ALS 8.3.2	Windows 7 64-bit workstation	$7,000	Octopus + FIJI	[1024, 1024, 1024	no GPU **310** w/ GPU **105**	[2048, 2048, 1024]	No GPU **1300** w/GPU **479**
APS 13-BM	Windows 7 64-bit workstation	$6,000	Gridrec + IDL	[1392, 1040, 900]	**282**	[2048, 2048, 1500]	**996**

- Many beamlines use large Linux clusters
  - Running multiple processes w/ MPI (complex; users cannot run at home)
- Even with those the reconstruction times for large datasets are 6 to 44 minutes

Conventional software is often not taking advantage of multiple cores!

- Windows task manager with existing Gridrec reconstruction software at our beamline
- Only 1 core being used
- Must be a better way!



New Approach: Single Multi-threaded Application

- tomoRecon: New reconstruction library.
  - Runs multiple slices simultaneously, each in its own thread (and core)
  - All run in a single process, much simpler than MPI
  - Runs on a single workstation
- Uses Gridrec for actual reconstruction
  - Very fast FFT-based reconstruction, already used at many sites
  - With appropriate singogram padding gives identical results to conventional filtered back-projection (F. Marone et. al SPIE 2010)

Software needs for multi-threading

- Applications written with multiple threads require a support library that provides:
  - Support for creating and operating on threads
  - Support for mutexes to prevent conflicts when threads need to access shared data
  - Support for a message passing system for passing data between threads
  - Support for events for signaling between threads
  - Support for date and time operations
- Ideally this support should operate transparently across multiple operating systems

EPICS libCom

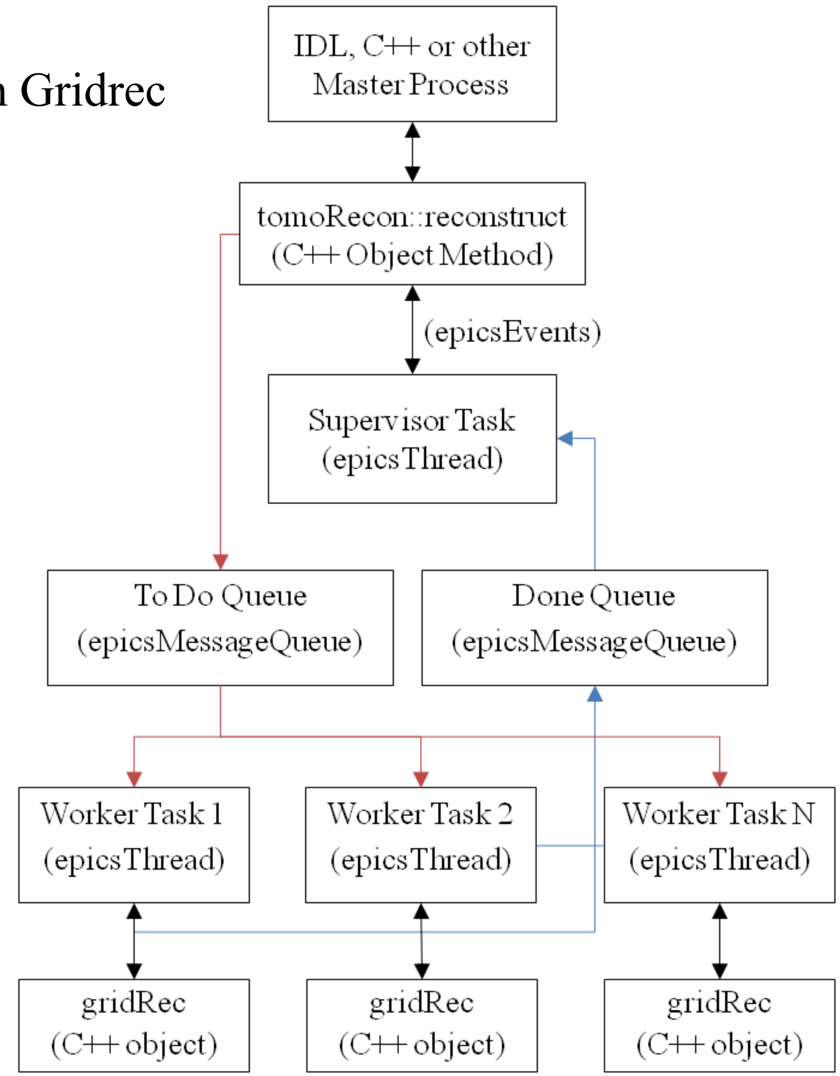
- EPICS is a control system toolkit in use at many synchrotrons both to run the accelerator and the beamlines
- EPICS has exactly the cross-platform library required for support for multi-threaded applications
- Run on Linux, Windows, Mac OSX, many others
- tomoRecon uses libCom

EPICS API	Functions used
epicsThread	epicsThreadCreate, epicsThreadGetNameSelf
epicsMessageQueue	epicsMessageQueueCreate, epicsMessageQueueTrySend, epicsMessageQueueTryReceive, epicsMessageQueueReceive, epicsMessageQueueDestroy
epicsEvent	epicsEventCreate, epicsEventSignal, epicsEventWait, epicsEventDestroy
epicsMutex	epicsMutexCreate, epicsMutexLock, epicsMutexUnlock, epicsMutexDestroy
epicsTime	epicsTimeGetCurrent, epicsTimeDiffInSeconds, epicsTimeToStrftime

tomoRecon C++ class

- Single C++ class called tomoRecon
- 545 lines of code, on top of ~800 lines of code in Gridrec

```
class tomoRecon {
public:
  tomoRecon(tomoParams_t *pTomoParams,
            float *pAngles);
  ~tomoRecon();
  virtual int reconstruct(int numSlices,
                          float *center,
                          float *pInput,
                          float *pOutput);
  virtual void supervisorTask();
  virtual void workerTask(int taskNum);
  virtual void sinogram(float *pIn,
                        float *pOut);
  virtual void poll(int *pReconComplete,
                    int *pSlicesRemaining);
```



tomoRecon Work Flow

- Reconstructs a set of slices. Passed
  - Number of slices,
  - Array containing the rotation center for each slice, and
  - Pointer to normalized 3-D input data array
  - Pointer to reconstructed output 3-D data array.
- Sends one message for each pair of slices to the worker tasks through the “To Do” message queue
  - Reconstructs two slices per message because Gridrec reconstructs two slices at once, one in the real part of the FFT, and the other in the imaginary part.
  - Messages contain pointers to the input and output data locations, and the rotation center to be used for that pair of slices.
- Worker tasks compute the sinograms from the normalized input data, and then reconstruct using Gridrec.
  - When reconstruction complete the worker task sends a message to the supervisor task via the “Done Queue” message queue indicating that those slices are done.
  - The worker task then reads the next message from the “To Do” queue and repeats the process.
- When supervisor task has received messages for all slices it sets a flag indicating that the entire reconstruction is complete.
- Supervisor and worker threads then wait for another event signaling either:
  - Another reconstruction should begin, or
  - tomoRecon object is being deleted, and the threads should exit.

tomoRecon::sinogram

- Takes the log of data (except for fluorescence tomography data).
- Optionally does secondary I_0 normalization
  – Uses average of values (typically air) at the start and end of each row of the sinogram.
  – Produces more accurate attenuation values when the beam intensity is changing with time, or changing between the sample in and out positions due to scintillator effects.
- Optionally does ring artifact reduction.
  – Computes the difference between the average row of the sinogram and the low-pass filtered version of the average row.
  – Difference is used to correct each column of the sinogram.
  – RingWidth specifies the size of the low-pass filtering kernel
- This is a C++ “virtual function”
  – Can create a derived class that re-implements this function to do sinograms differently without modifying tomoRecon

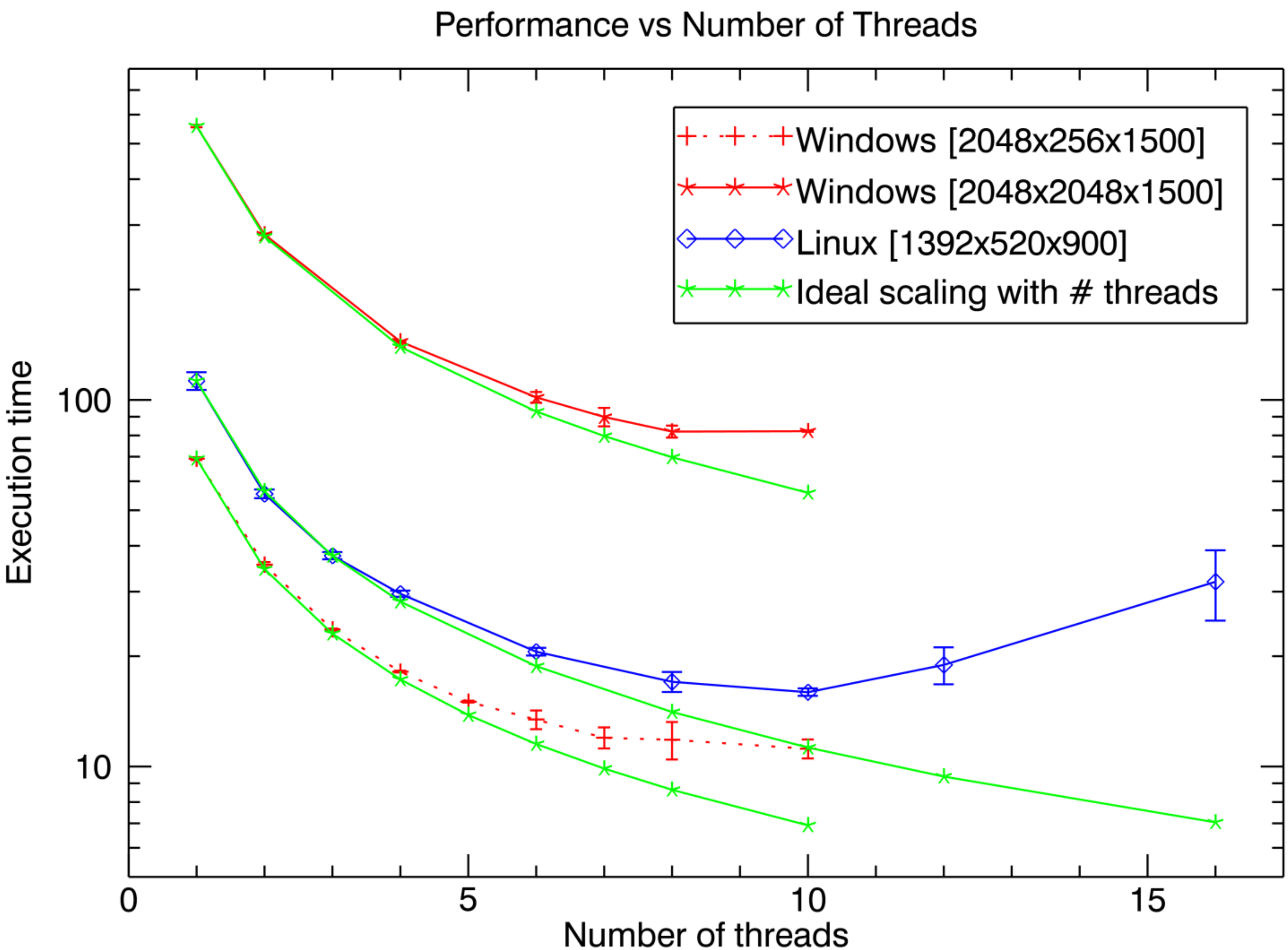
Gridrec

- New version of the Gridrec code.
  - Original Gridrec written in C, used static C variables to pass information between C functions.
  - Not thread-safe, each thread needs its own copy of such variables.
  - Gridrec was re-written in C++, with all such variables placed in private class member data.
- Gridrec was originally written to use the Numerical Recipes FFT functions four1 and fourn.
  - Previously user wrapper routines that maintained the Numerical Recipes API, but used FFTW, which is very high performance FFT library. Those wrapper routines were also not thread-safe, and they copied data, so were somewhat inefficient.
  - New version of Gridrec has been changed to use the FFTW API directly, it no longer uses the Numerical Recipes API.

Performance Tests

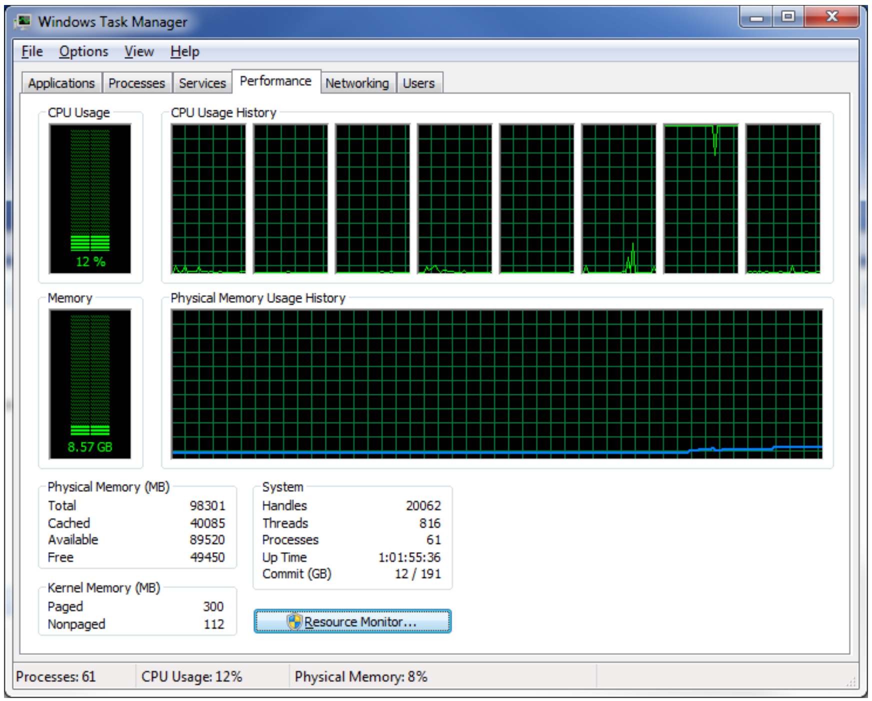
- Performance tests were done to determine:
  1. Reconstruction time as a function of number of threads for in-memory data
  2. Reconstruction time as a function of dataset size for in-memory data
  3. Reconstruction time, including reading the input file and writing the output file, as a function of the number of slices reconstructed in each call to tomoRecon.
- Tests done on Windows 7 tower workstation and a Linux rackmount server

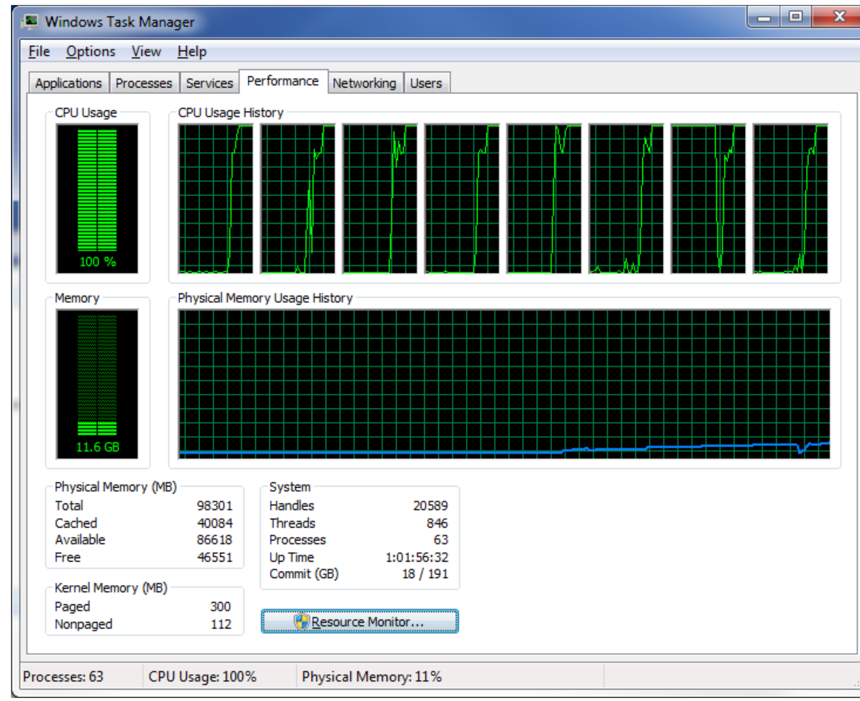
Computer type	Dell Precision T7500	Penguin Relion 1751 Server
Operating system	Windows 7 64-bit	Linux, Fedora Core 14, 64-bit
CPU	Two quad-core Xeon X5647, 2.93 GHz (8 cores total)	Two quad-core Xeon E5630, 2.53 GHz hyperthreading, (8 cores, 16 threads total)
System RAM	96 GB	12 GB
Disk type	Two 500 GB 15K RPM SAS disks RAID 0	Three 300 GB 15K RPM SAS disks No RAID.
Approximate cost	$6,000	$5,000



- Close to ideal performance up to 5-6 threads
- Windows does not improve much after 7 threads
- Linux has a minimum at 10 threads, then increases

We' re now using all the cores!





tomoRecon with 1 thread

tomoRecon with 8 threads

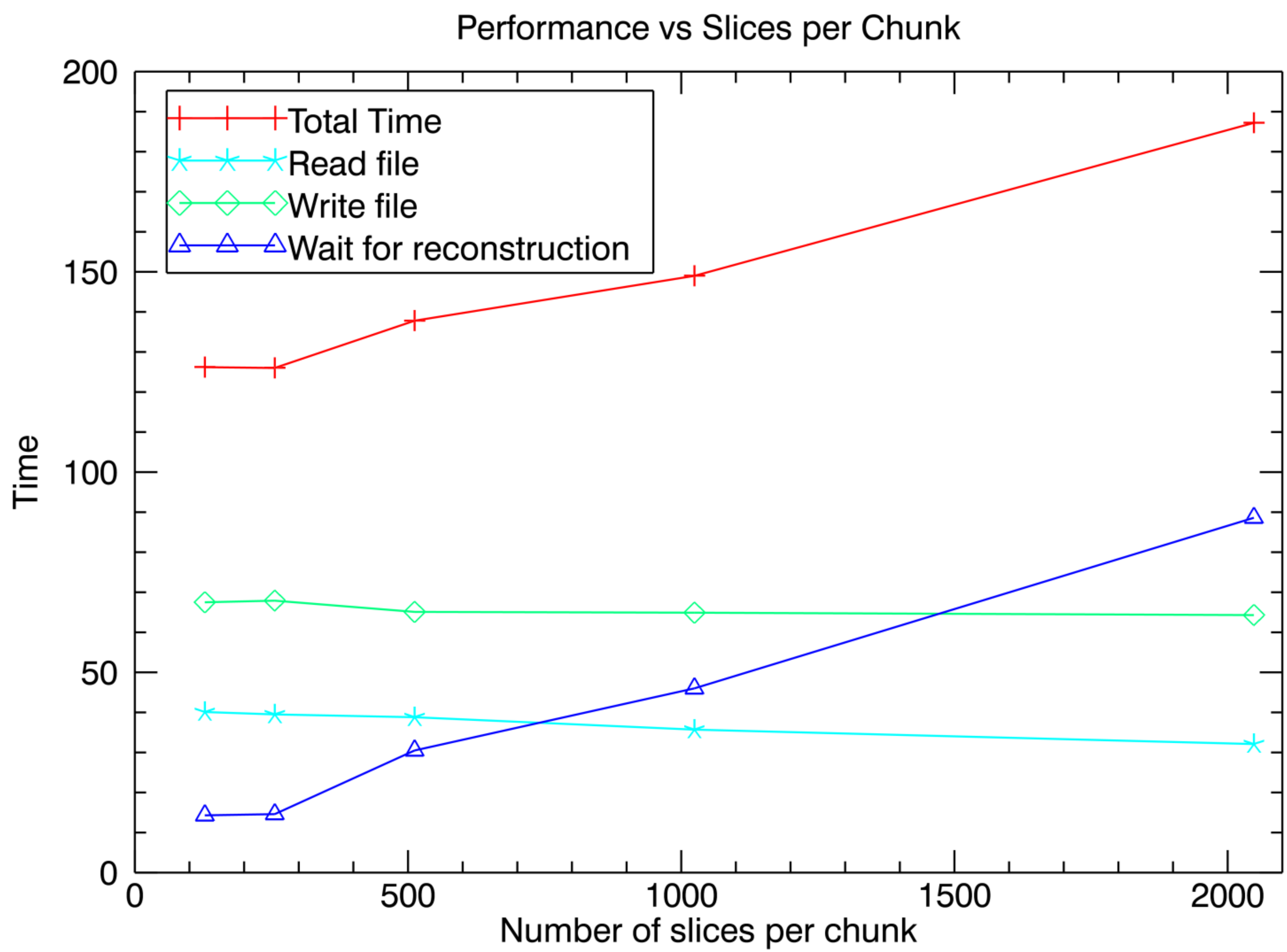
Performance as a function of dataset size

Dataset size	Windows (8 threads)	Linux (12 threads)
[696, 520, 720]	**4.13 ±0.02**	**4.57 ±0.03**
[1392, 1040, 900]	**24.8 ±0.5**	**37.2 ±3.7** (measured 18.6 for 520 slices, insufficient memory for all slices)
[2048, 2048, 1500]	**83.1 ±3.0**	**127.9 ±29.0** (measured 16.0 ±3.6 for 256 slices, insufficient memory for all slices)

- Time for large [2048, 2048, 1500] dataset is only 83 seconds on Windows with 8 threads.
- 6-40 times faster than existing computing clusters However, does not include the time to read the input file and write the output file.

Performance Including File I/O vs Chunk Size

- tomoRecon runs reconstructions in the "background" in supervisor and worker threads
- Can thus overlap reading and writing files with reconstruction using 2 buffers, B1 and B2 in a loop.
- Each pass through loop reconstructs a set of slices (=chunk)
  - First time read B1 and start it reconstructing in background
  - Then read B2 and wait for the B1 reconstruction to complete.
  - Immediately start reconstructing B2, then write the B1 reconstruction and read the next B1 input.
  - Waits for the B2 reconstruction to complete, starts B1 reconstructing etc.
  - In the optimal case reconstruction of B2 has just completed when the file writing of the previous B1 and reading of the next B1 complete.
  - If so, process is entirely limited by the file I/O and the reconstruction does not slow the process down at all!

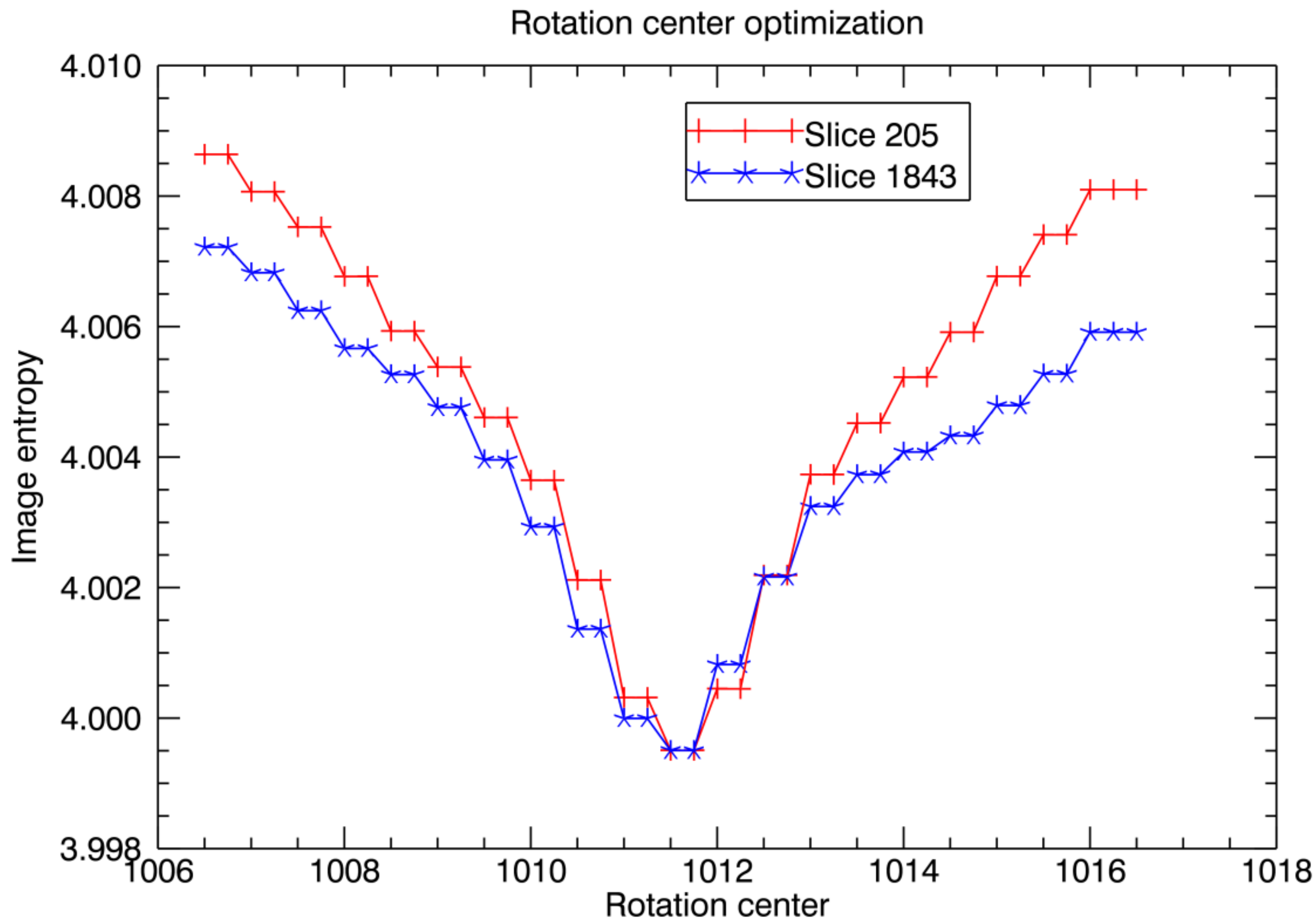


Performance vs Slices per Chunk
Total Time
Read file
Write file
Wait for reconstruction
Time
Number of slices per chunk
0
50
100
150
200
0
500
1000
1500
2000

Performance including file I/O vs chunk size

Slices per chunk	Number of chunks	Read time	Write Time	Wait for reconstruction	Total Time	Total Required RAM (GB)
2048	1	32.1	64.3	88.6	187.2	60
1024	2	35.7	64.9	46.0	149.0	44
512	4	38.8	65.1	30.5	137.8	32
256	8	39.5	67.9	14.6	126.0	16
128	16	40.1	67.5	14.3	126.2	12

- Single Windows workstation using tomoRecon can do complete [2048,2048,1500] reconstruction, including file I/O, in ~2 minutes.
- 4-20 times faster than existing systems, including large clusters



- Find optimum value of rotation axis to sub-pixel by entropy minimization
- tomoRecon reconstructs same slice N times using different rotation center for each N
- Done in parallel in multiple threads
- 4 seconds to search 41 center positions, +- 5 pixels in 0.25 pixel steps
- Do slices near top of sample and bottom of sample to correct for any slight misalignment of rotation axis and CCD columns

Future plans

- Integrate pre-processing (normalization to dark and flat field, zinger removal) either into tomoRecon or another C++ threaded package, or IDL with new GPU library.
- Make an ImageJ front-end to tomoRecon by using the Java Native Interface (JNI)
- Reconstruction is only the first step in tomography data processing. Other steps could benefit from the same architecture used here if the code is or can be written in C or C++.

Conclusions

- tomoRecon allows single workstation to do large-scale reconstructions previously limited to clusters
- Fully utilizes large memory and multiple cores of modern computers
- tomoRecon source code and pre-built libraries for Linux and Windows, and Mac available here:

  http://cars.uchicago.edu/software/epics/tomoRecon.html
- IDL front-end software available here:

  http://cars.uchicago.edu/software/IDL/tomography.html

Thanks for your attention!!!