Microsoft® Research

FacultySummit 2011

Cartagena, Colombia | May 18-20 | In partnership with COLCIENCIAS

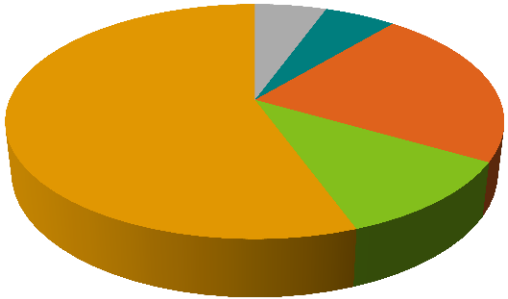
Tools of the trade: Technical Computing on the OS … that is not Linux!

Or how to leverage everything you've learned, on a Windows box as well

Sean Mortazavi & Felipe Ayora

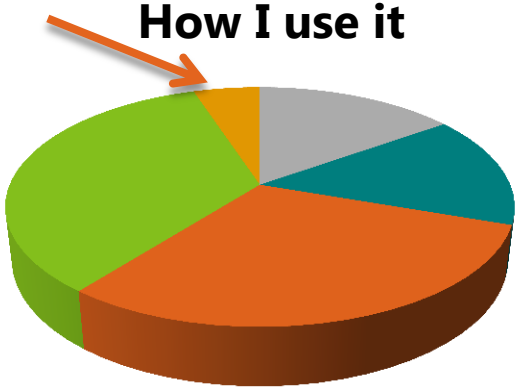
Typical situation with TC/HPC folks

Why I have a Windows box



- It was in the office when I joined
- IT forced me
- I couldn't afford a Mac
- Because I LIKE Windows!
- It's the best gaming machine

How I use it



- Outlook / Email
- PowerPoint
- Excel
- Gaming
- Technical/Scientific computing

Note: Stats completely made up!

The general impression

- “Enterprise community”
- Guys in suits
- Word, Excel, Outlook
- Run prepackaged stuff

- “Hacker community”
- Guys in jeans
- Emacs, Python, gmail
- Builds/runs OSS stuff

Common complaints about Windows

- I have a Windows box, but Windows …
  - Is hard to learn… ☹
  - Doesn't have a good shell ☹
  - Doesn't have my favorite editor ☹
  - Doesn't have my favorite IDE ☹
  - Doesn't have my favorite compiler or libraries ☹
  - Locks me in ☹
  - Doesn't play well with OSS ☹
  - …. ☹
- In summary:

☹

ஜ (More like )

My hope …

- I have a Windows box, and Windows …
  - Is easy to learn… ☺
  - Has excellent shells ☺
  - Has my favorite editor ☺
  - Supports my favorite IDE ☺
  - Supports my compilers and libraries ☺
  - Does not lock me in ☺
  - Plays well with OSS ☺
  - …. ☺
- In summary:

(or at least 😐)

How?

- Recreating a Unix like veneer over windows to minimize your learning curve
- Leverage your investment in know how & code
- Showing what key codes already run natively on windows just as well
- Kicking the dev tires using cross plat languages

Objective is to:

Help you ADD to your toolbox,
not take anything away from it!

At a high level…

- Cygwin
- SUA
- Windowing systems
- Standalone shell/utils

"The Unix look & feel"

- IDE's
- Editors
- Compilers / languages / Tools
- make
- Libraries

General purpose development

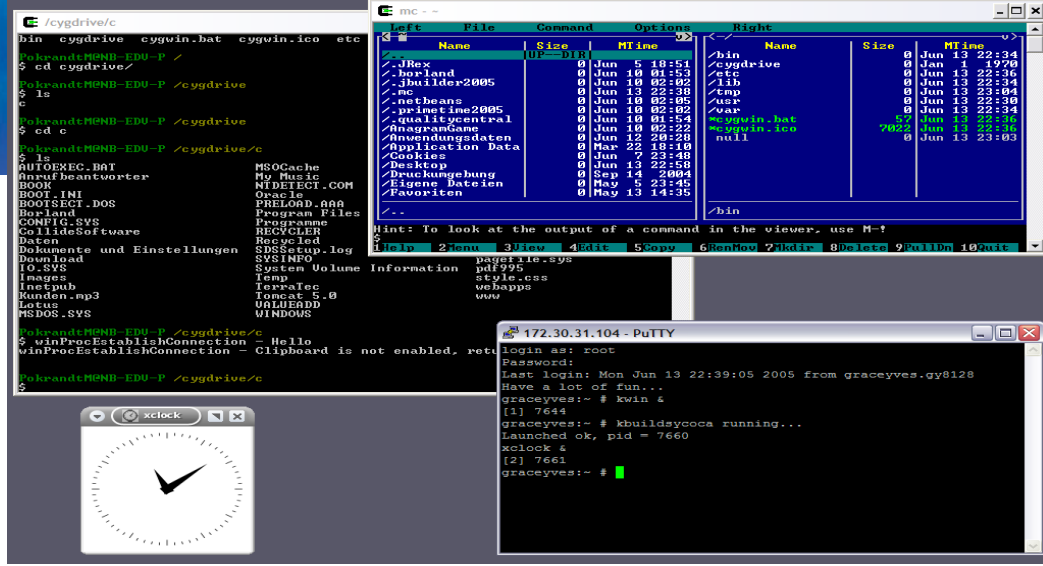
- CAS environments

Dedicated CAS / IDE's

And if there is time, a couple of demos…

Cygwin

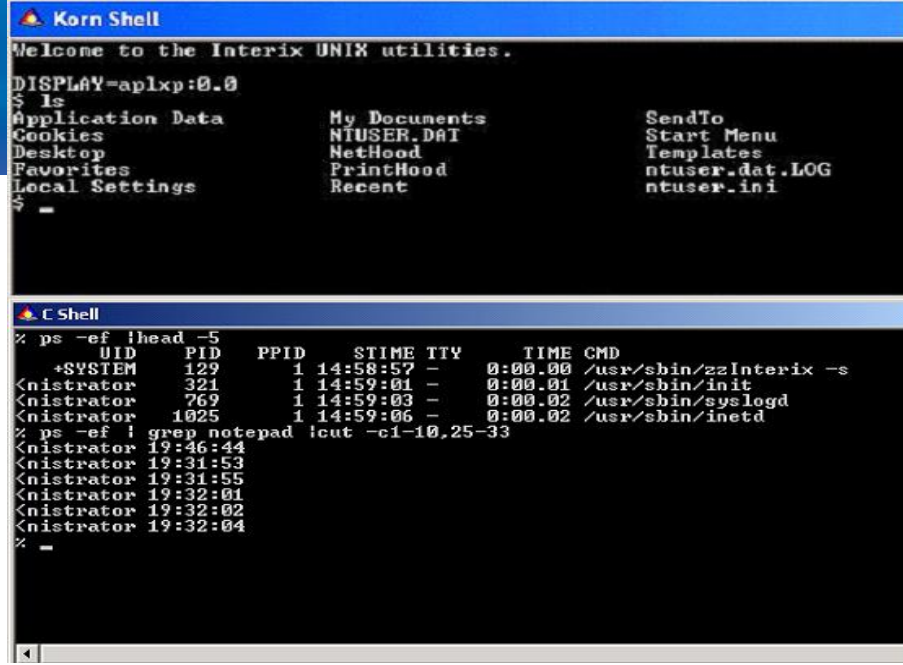
- What is it?
  - A Unix like environment for Windows. Native integration of familiar Unix tools & apps built from source for Windows.
- How does it work?
  - POSIX support is provided by the cygwin.dll library which enables code migration w minimal changes
  - The usual shells, utilities, etc are compiled as native Windows binaries against cygwin.dll
- What it's not
  - Cygwin doesn't provide emulation of Unix programs under windows – apps must be built from src
- Integration
  - All std tools, shells, mounts, file conversions, symlinks, ACL's, various langs + gcc for windows, ssh, telnet, ftp, …
  - You can use win32 API's *and* POSIX



- Sample ports
  - Openoffice, Sun Java, …
- License, who
  - GPL v2, redhat
- Useful links
  - www.cygwin.com

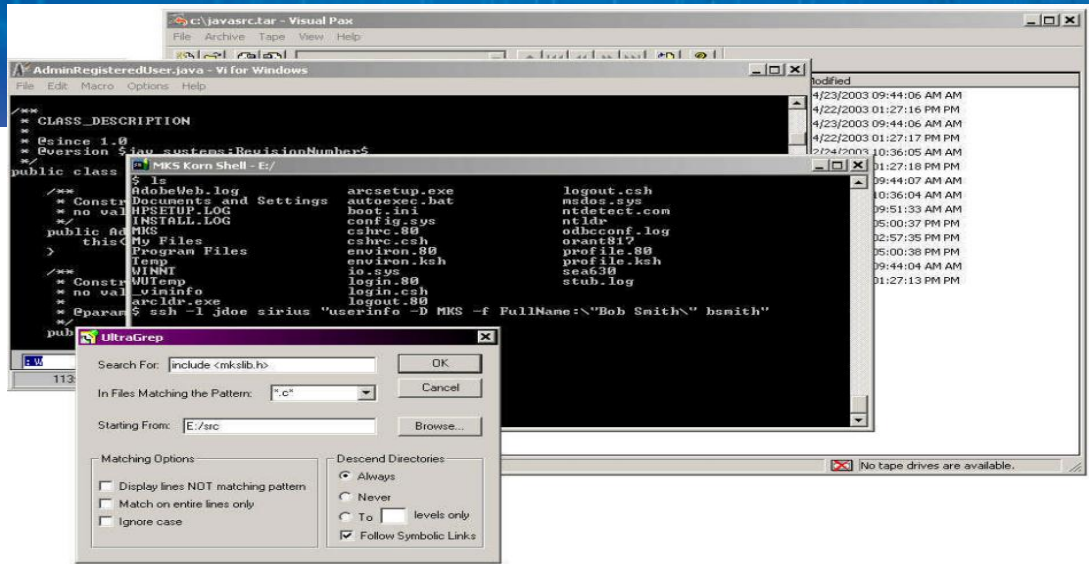
SUA
Subsystem for Unix Applications

- What is it?
  - A Unix-like environment for Windows as a subsystem. Native integration of familiar Unix tools & apps built from source for Windows.
- How does it work?
  - Similar to Cygwin, but implemented as subsystem
  - All tools, shells, utils are built from source, link to Windows C runtime. Support for msft compilers via wcc wrapper.
- What it's not
  - SUA doesn't provide emulation of Unix programs under windows
- Integration
  - All std tools/shells + better NFS, Oracle/MSFT SQL, AD/user mgmt support
  - With later version can mix win32 & posix calls



- License, who
  - Free with versions of Windows, MSFT
- Useful links:
  - http://bit.ly/fTVRtu for the SDK tools
  - http://bit.ly/jwGCpD unix/windows dictionary
  - http://bit.ly/kK3f15 SUA community

MKS Toolkit



- What is it?
  - Another Unix-like environment for Windows . Native integration of familiar Unix tools & apps built from source for Windows.
- How does it work?
  - Native ports of unix shell/utils
- What it's not
  - MKS doesn't provide emulation of Unix programs under windows
  - MKS is not free
- Integration
  - Posix support DLL like cygwin
  - Good Enterprise support for AD, users, pwd sync, WMI, …
  - Best of the bunch for sysadmin'ing heterogeneous farms

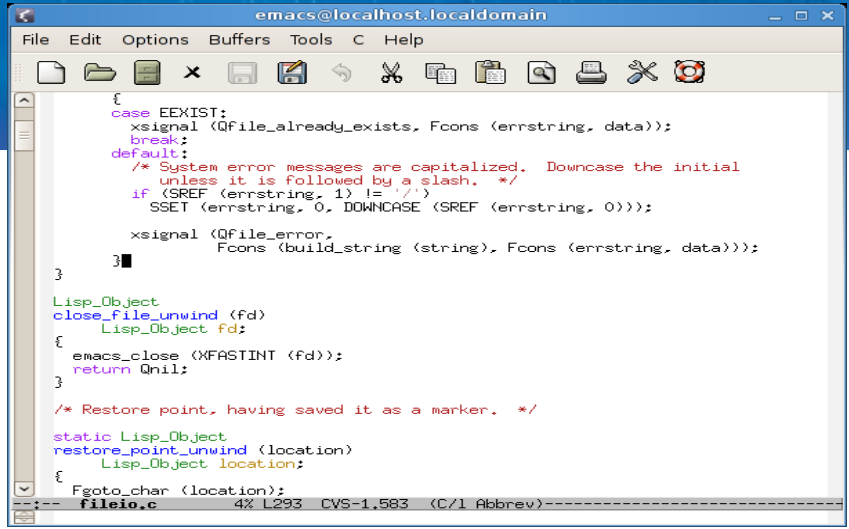
- License, who
  - Proprietary, MKS
- Useful links:
  - http://mkssoftware.com general
  - http://www.mkssoftware.com/products/tk/commands.asp?product=tkdev developers

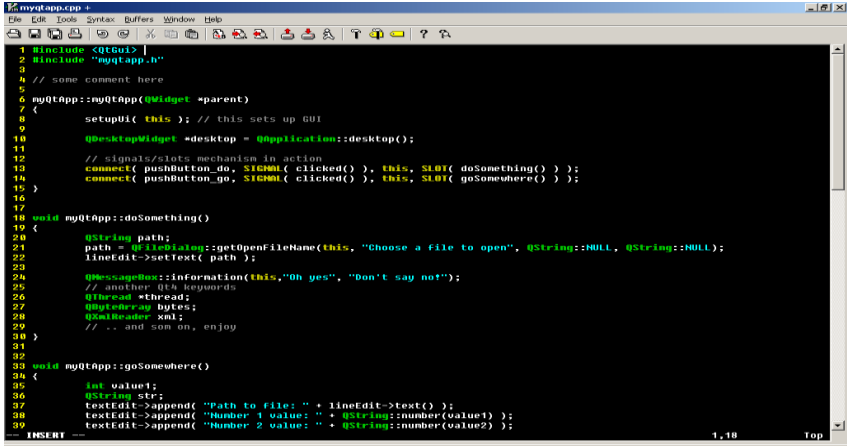
Shells & Utilities options

- All common shells are available on Windows:
  - sh, bash, csh, tcsh, zsh, kshell, …
- Getting them - Option 1: “Distros”
  - MKS
  - SUA
  - Cygwin
  - MSYS
  - …
- Getting them - Option 2: “Just the basics please”
  - GnuWin – windows versions of gnutools
    - Native ports of 150+ utils – using mingw / MSVC, no emulation
    - Distributed w gnuemacs, KDE, …
    - GPL

Editors

- The classics
  - Vi
  - Vim
  - Emacs
- The newer batch
  - Notepad2
  - Notepad++
  - E (Textmate)
  - TotalEdit
  - UltraEdit
  - Pspad
  - EditPlus
  - …
- Emulation support in Visual Studio
  - ViEmu
  - VsVim
  - Emacs



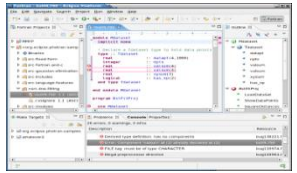


Demo: Shells

- Installation & usage of Unix shells / utilities
- PowerShell: the new msft shell – mixing & matching shells
- Vi / Emacs

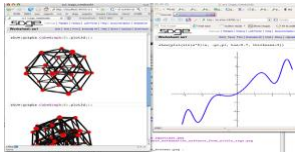
IDE examples

- Eclipse
  - Started a Java env, now w C++ and various other plug-ins

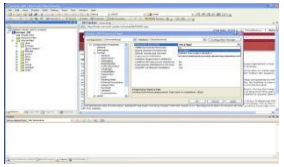


- IPython
  - Interactive Python REPL w support for parallel computing

- Sage
  - Symbolic math IDE w Python as the scripting language



- Visual Studio
  - Various languages w support for cross plat compilers



- RevoAnalytics R
  - A complete R development environment w debugging and visualization

...and of course emacs ☺

Build environments

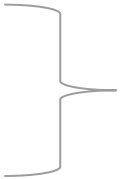
- MKS / CygWin / SUA
- MingW
- Cross-plat compilers
- Scripts, make, nmake, Cmake, …
- /, vs \, File vs file, drive names, …

```
# first we add the executable that generates the table
add_executable(MakeTable MakeTable.cxx)

# add the command to generate the source code
add_custom_command (
  OUTPUT ${CMAKE_CURRENT_BINARY_DIR}/Table.h
  COMMAND MakeTable ${CMAKE_CURRENT_BINARY_DIR}/Table.h
  DEPENDS MakeTable
  )

# add the binary tree directory to the search path for
# include files
include_directories( ${CMAKE_CURRENT_BINARY_DIR} )

# add the main library
add_library(MathFunctions mysqrt.cxx ${CMAKE_CURRENT_BINARY_DIR}/Table.h  )
```

The Cmake crossplat build utility

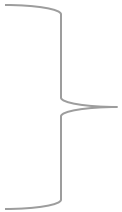
Languages, Compilers/interpreters

- Classics
  - C/C++
  - Fortran
- Newer on the block
  - C#/Java
  - Python
  - R
  - F#
- Dedicated CAS langs
  - M
  - Mathematica
  - Maple
  - …

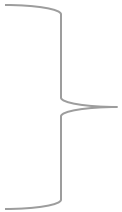


- C: gcc, VC++, Intel*, clang, …
- Fortran: Intel*, PGI*, …

* generally the best FP optimizations



- JVM, CLR langs
- CPython, IronPython, Pypy, Jython, …
- Revolution R, R-Studio



- IDE + language combos

C#/F# run on linux/macos via Mono

Library/runtime Examples

- Boost
- Python: SciPy, Numpy, scikits, …

- From text to image processing to large graphs & MPI
- Numeric & Scientific libraries

- MKL
- IMSL
- NAG
- Visual Numerics
- LibFlame
- ATLAS

- Highly optimized Math libraries for native & .Net number crunching

- MPI, OpenMP, CUDA, …

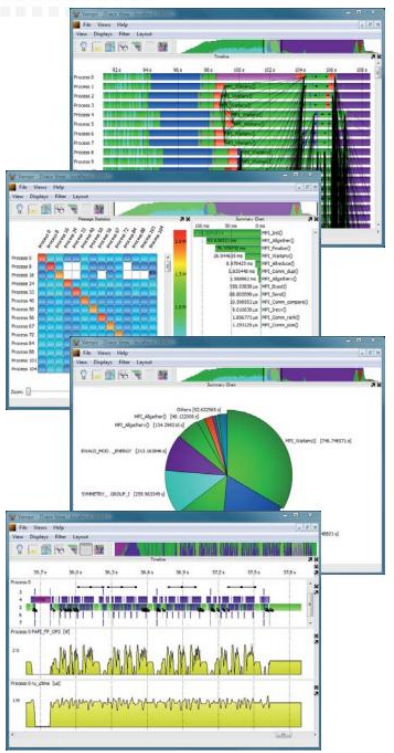
- Optimized || libraries for multicore, cluster and GPUs

- NetCDF, HDF5, FITS, …

- OS & language neutral scientific data formats & libraries

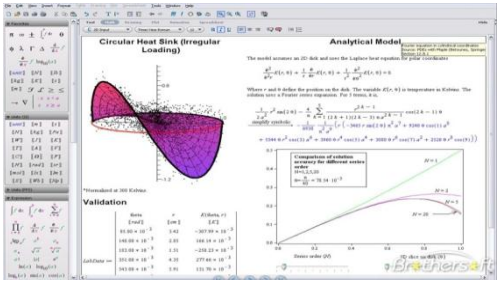
Debugging, Profiling, Analysis

- VS plug-ins from Intel, PGI, Allinea (MPI)
- VS plug-in for Python/mpi4py
- MPI Tracing
  - JumpShot
  - Vampir (native windows port)
- ETW (“DTrace”)
  - Event Tracing for Windows
- Marmmot for VS
  - MPI call / param analysis (runtime)
- ISP
  - MPI program verification

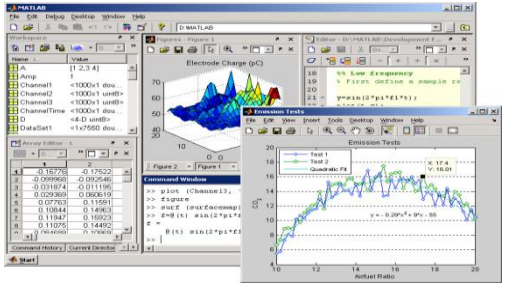


Vampir for Windows from U Dresden

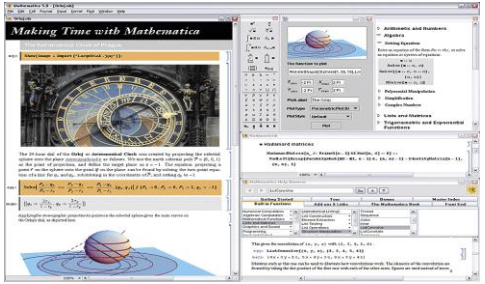
Computer Algebra Systems



Maple *

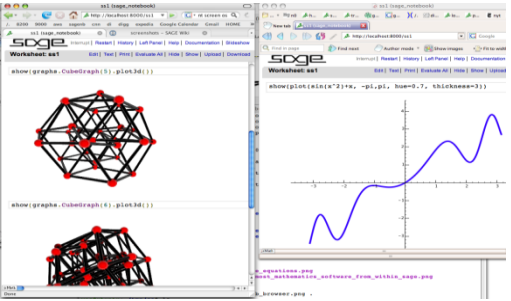


MATLAB *

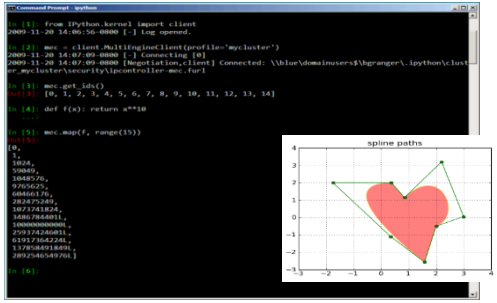


Mathematica *

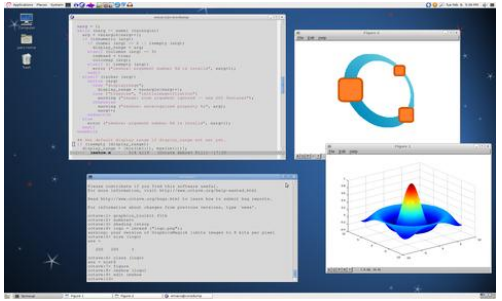
Commercial



Sage



IPython *



Octave

Free/OSS

* Can parallelize using Windows HPC

Now that you're here… stuff you really should try!

- Visual Studio
- F#
- MPI.Net
- Python Tools for VS
- PowerShell
- CUDA

Visual Studio

- State of the art development environment
  - Development, debugging, profiling, life-cycle mgmt, …
- Technical Computing related
  - Best host for key compilers: C++, Fortran, PGI
  - Free full featured Python dev plug-in
  - Best host for AMD, Nvidia plug-ins: see Nsight
  - Support for Vim & Emacs ☺
- Academic / hobbyist related
  - Essentially free to schools via www.dreamspark.com program
  - Free to startups via www.bizspark.com program

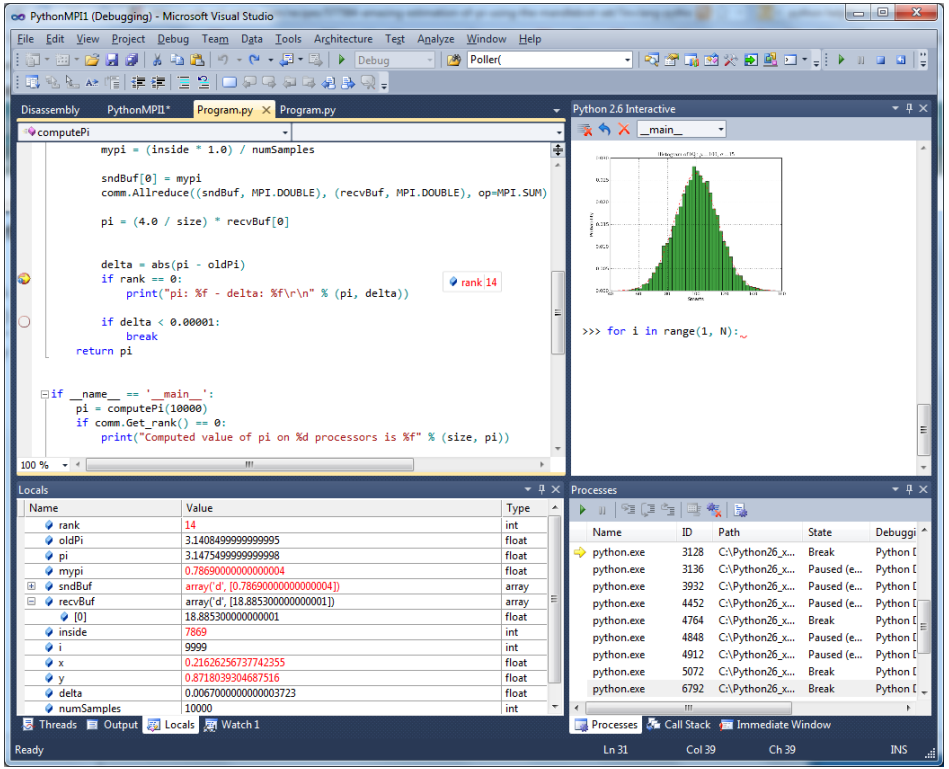
F#

- Multi-paradigm language
- Succinct & powerful
- Interactive mode
- Runs on MacOS & Linux too!
- Free / open source
- Check out “Units of Measure”, async

- See Christophe’s **demo** at the demofest!

Python Tools for VS

- Free & Open source plug-in for writing Python code
- Intellisense, browsing, …
- Standard REPL + IPython REPL
- Cluster support: MPI & IPython
- Debugging
- Profiling
- CPython, IronPython, Jython, …
- Soon: Big Data, Big Compute support



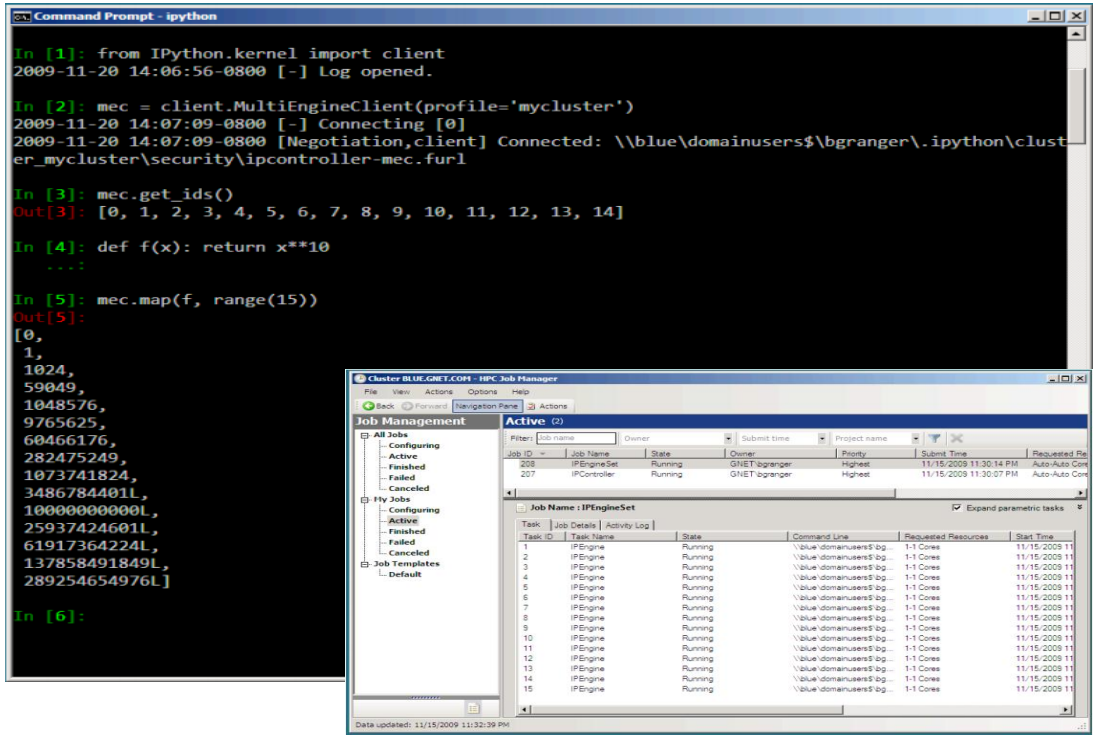
http://codeplex.pytools.com

Demo

- Python Tools for Visual Studio
  - Intellisense
  - Profiling
  - Cluster MPI debugging

Noteworthy: IPython

- Interactive computing using Python
  - Advanced REPL with History, completion, …
  - Capture ‘var = !ls –la’
  - Inline images
- Interactive || computing
  - Specify cluster headnode,
  - Start # of desired engines
  - Compute!
- Included in all major Python distro’s
- Open source & available on Windows, Linux, Mac



Fernando Perez & Brian Granger

Demo

Why Python

- Well suited to Technical & Scientific Computing
  - Isn't it interpreted? (and slow?!)
  - Interactive by design
  - Easily mix in native code, even CUDA, MPI, … see Cython, swig, ctypes
- Easy ramp up, yet powerful language
  - From CS101 to PhD thesis
- Incredibly rich ecosystem of high quality libraries
  - Web, numerics, symbolics, Bio, astronomy, …
  - 2D, 3D viz, …. See scipy, scikits
- Healthy developer eco-system
  - #3 most popular in some surveys
  - PyCon is the main conference
- Various implementations
  - CPython,IronPython, Jython, PyPy, …
- Free, open source w quality distro's providing support
  - BSD like license
  - Enthought Python Distro, ActivePython, …

MPI.Net : A high performance wrapper for MPI

C vs C#: gather cluster hostnames / sort / print

```
Communicator comm = Communicator.world;
string[] hostnames =
  comm.Gather(MPI.Environment.ProcessorName, 0);
if (Communicator.world.Rank == 0) {
  Array.Sort(hostnames);
  foreach (string host in hostnames)
    Console.WriteLine(host);
}
```

```
int size, rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

char name[MPI_MAX_PROCESSOR_NAME];
int resultlen;
MPI_Get_processor_name(name, &resultlen);

int *rbuf = (int*)malloc(sizeof(int) * size);
MPI_Gather(&resultlen, 1, MPI_INT, rbuf, 1, MPI_INT, 0, MPI_COMM_WORLD);

int *rcounts = (int*)malloc(sizeof(int) * size);
int *rdispls = (int*)malloc(sizeof(int) * size);
int cnt = 0;
for (int i = 0; i < size; i++) {
  rcounts[i] = rbuf[i]+1;
  cnt += rcounts[i]+1;
  if (i) rdispls[i] = rdispls[i-1]+rbuf[i]+1;
  else rdispls[i] = 0;
}
char *rnamebuf = (char*)malloc(cnt * sizeof(char));
MPI_Gatherv(name, resultlen, MPI_CHAR, rnamebuf, rcounts, rdispls, MPI_CHAR, 0,
   MPI_COMM_WORLD);
if (rank == 0) {
  char **hostnames = (char**)malloc(size * sizeof(char*));
  for (int i = 0; i < size; ++i)
    hostnames[i] = rnamebuf + rdispls[i];
  qsort (hostnames, size, sizeof(char*), strcmp);
  for (int i = 0; i < size; ++i)
    printf("%s\n", hostnames[i]);
}
```

http://www.osl.iu.edu/research/mpi.net/

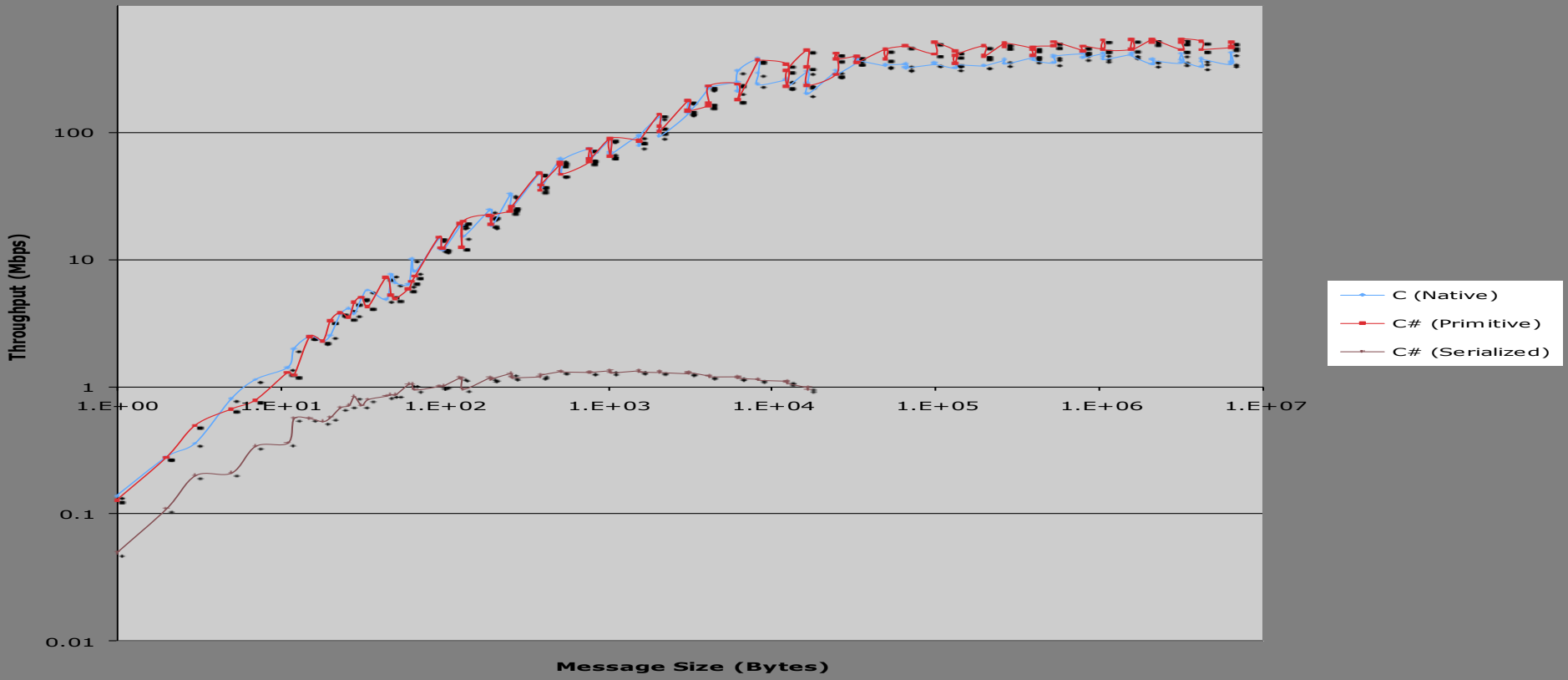
Optimizing MPI.Net performance

An optimal Send?

```
public void Send<T>(T value, int dest, int tag)
{
  if (HasMpiDatatype<T>()) {
    unsafe {
      fixed (T* valuePtr = &value) {
        Unsafe.MPI_Send(new IntPtr(valuePtr), 1,
                GetMpiDatatype<T>(), dest, tag, comm);
      }
    }
  } else {
    // Serialize and transmit
  }
}
```

C#	MPI
short	MPI_SHORT
int	MPI_INT
float	MPI_FLOAT
double	MPI_DOUBLE

NetPIPE Performance

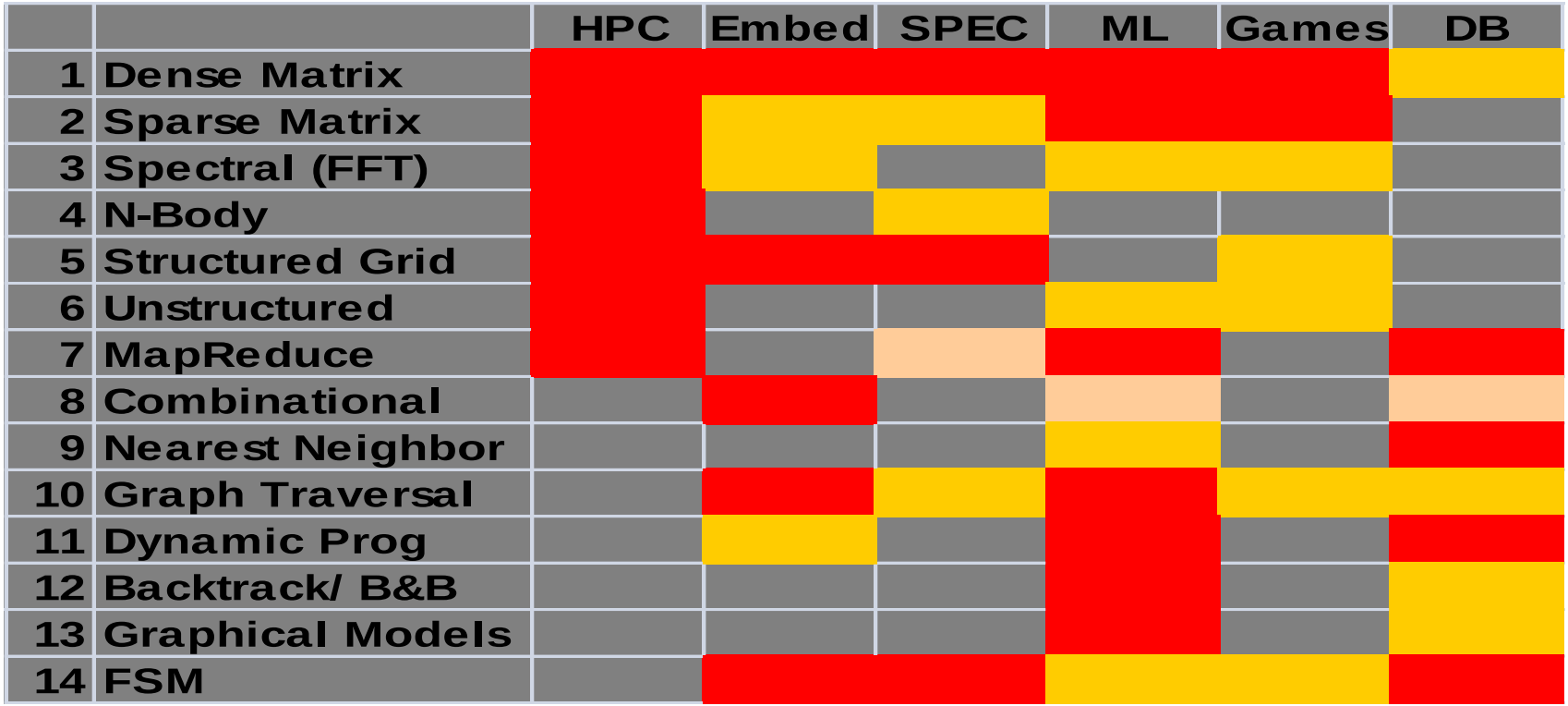


Parallel Dwarfs Project

- The “Dwarfs”
  - Colella’s original 7, now 13 kernels that encapsulate a large spectrum of computing workloads
- Parallel Dwarfs: Visual Studio solutions that implement || versions of the dwarfs (13k+ LOC)
  - Languages: C++, C#, (some F#)
  - Input files: small, medium, large
  - Parallelization technologies:
    - OpenMP, TPL, MPI, MPI.Net, (ClusterSOA, PPL soon)
  - Results gathering & plotting
    - Excel, JumpShot, Vampir
  - “Driver” for selecting & running the benchmarks
  - Open Source on codeplex.com

Dwarf Popularity[1]

> >



		HPC	Embed	SPEC	ML	Games	DB
1	Dense Matrix						
2	Sparse Matrix						
3	Spectral (FFT)						
4	N-Body						
5	Structured Grid						
6	Unstructured						
7	MapReduce						
8	Combinational						
9	Nearest Neighbor						
10	Graph Traversal						
11	Dynamic Prog						
12	Backtrack/ B&B						
13	Graphical Models						
14	FSM						

Source: "Future of Computer Architecture" by David A. Patterson

Scale: Nearly 300 combinations

- Foreach (managed, unmanaged)
  - Foreach (mpi, mpi.net, openmp, tpl, hybrid)
    - Foreach (input.small, input.medium,input.large)
      - Foreach (one..thirteenth dwarf)
        - Run, Trace
        - Plot Excel, Xperf
        - Plot Vampir, JumpShot
- Support for mixed models:
  - MPI + Openmp
  - MPI.Net + TPL
  - etc

Use the Parallel Dwarfs for:

- Comparing || technologies
- Comparing language features
- Benchmarking
- Best practices
- Starting templates

PS1> DwarfBench -Names SpectralMethod -Size medium -Platform managed -Parallel serial,tpl,mpi –PlotExcel

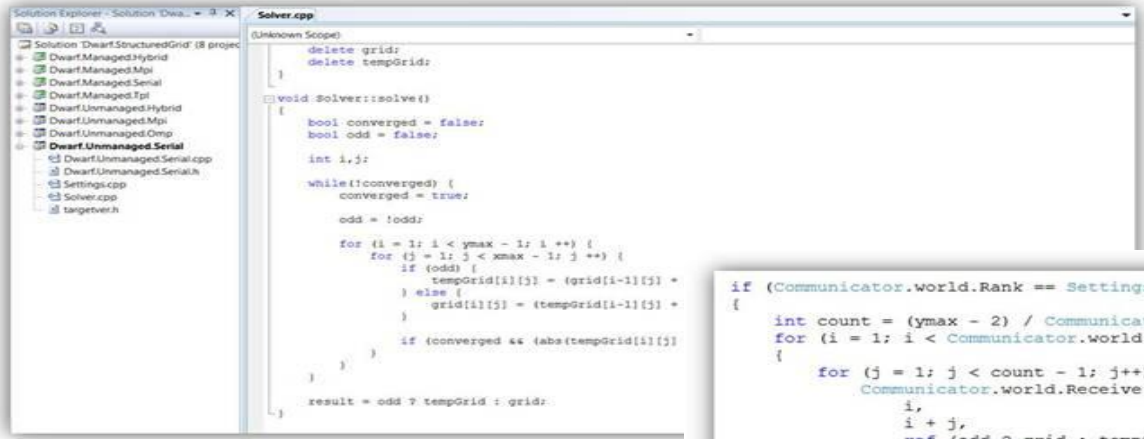
PS1> DwarfBench –Names DenseAlgebra -Size medium -Platform unmanaged,managed -Parallel mpi –PlotVmampir

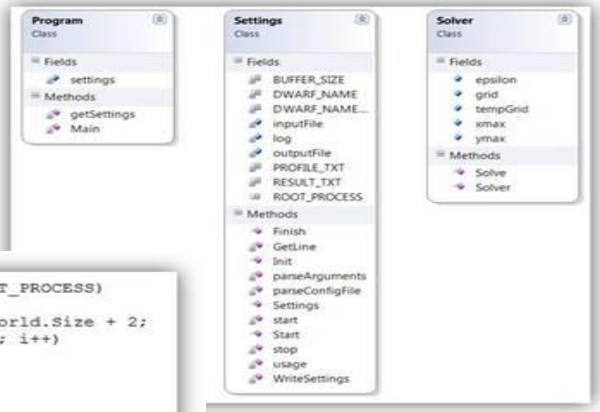
PS1> DwarfBench –Names *grid* -Size Large -Platform unmanaged -Parallel hybrid –PlotVmampir

Skip/Demo

13 Visual Studio Solutions

StructuredGrid code fragment using MPI.NET

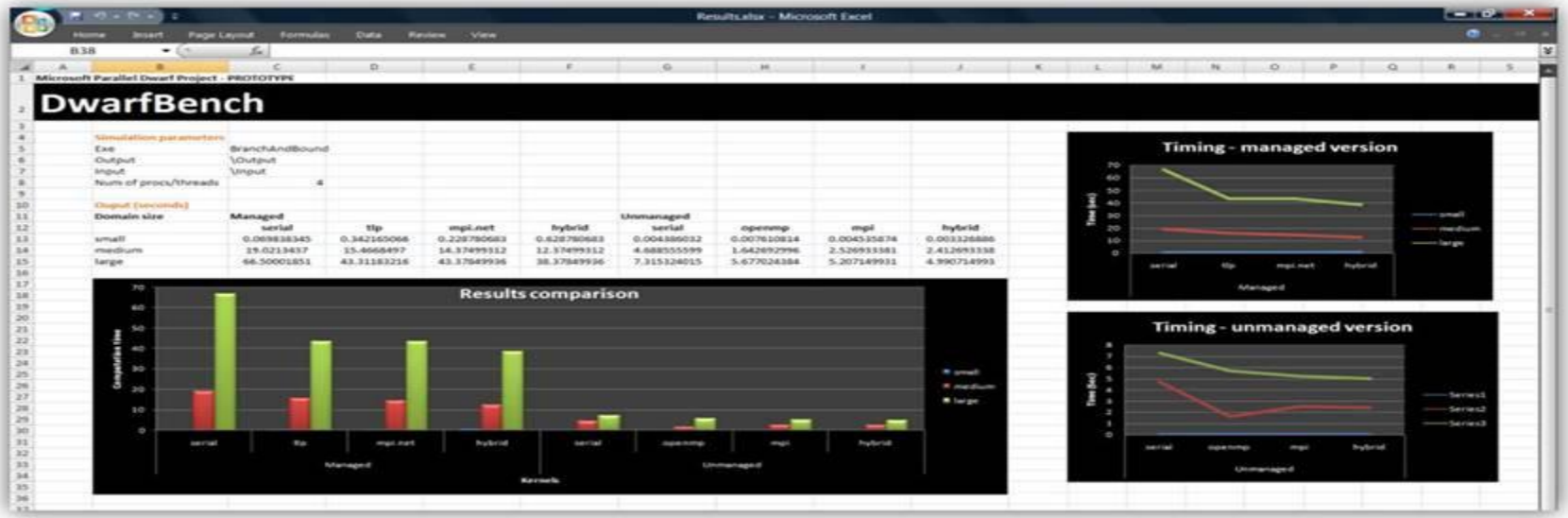





```
if (Communicator.world.Rank == Settings.ROOT_PROCESS)
{
    int count = (ymax - 2) / Communicator.world.Size + 2;
    for (i = 1; i < Communicator.world.Size; i++)
    {
        for (j = 1; j < count - 1; j++)
            Communicator.world.Receive(
                i,
                i + j,
                ref (odd ? grid : tempGrid)[(ymax - 2) %
                Communicator.world.Size + ((ymax - 2) /
                Communicator.world.Size) * i + j]);
    }

}
else
{
    for (i = 1; i < countRows - 1; i++)
        Communicator.world.Send(
            (odd ? grid : tempGrid)[i],
            Settings.ROOT_PROCESS,
            Communicator.world.Rank + i);
}

grid = (odd ? grid : tempGrid);
```

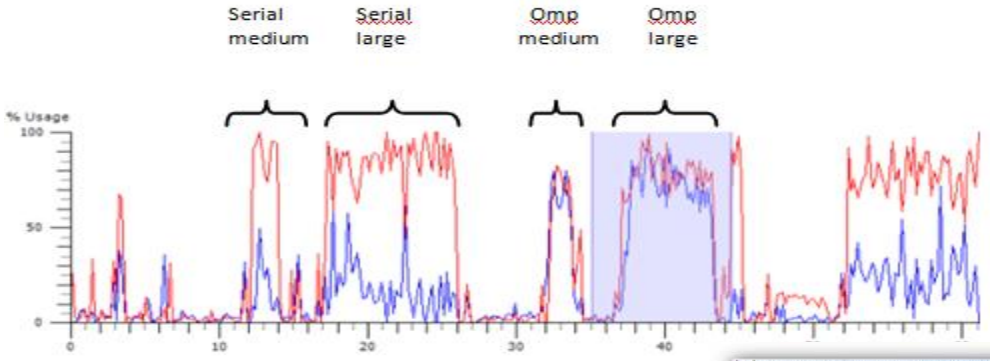
Skip/Demo

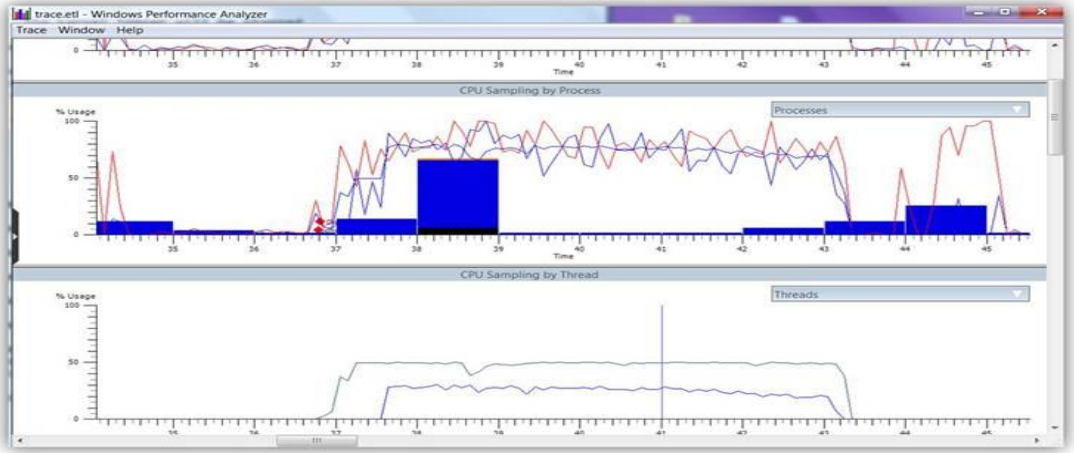
Results Summary: -PlotExcel



Skip/Demo

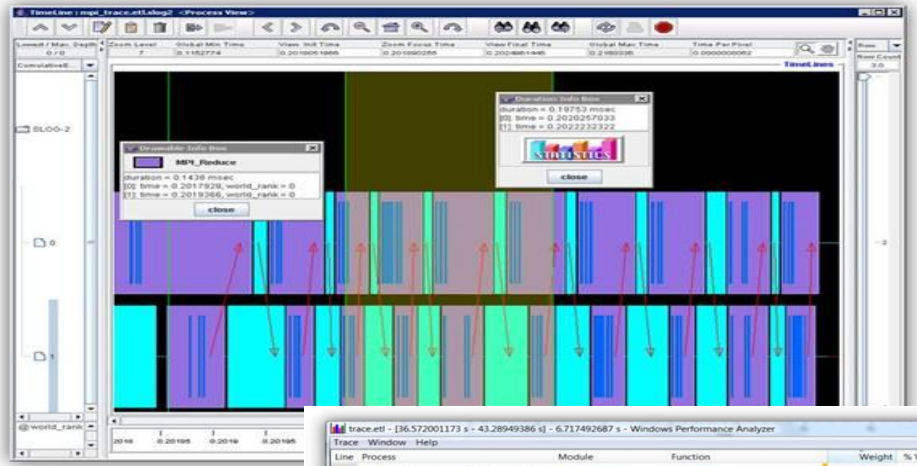
Xperf: -PlotXperf

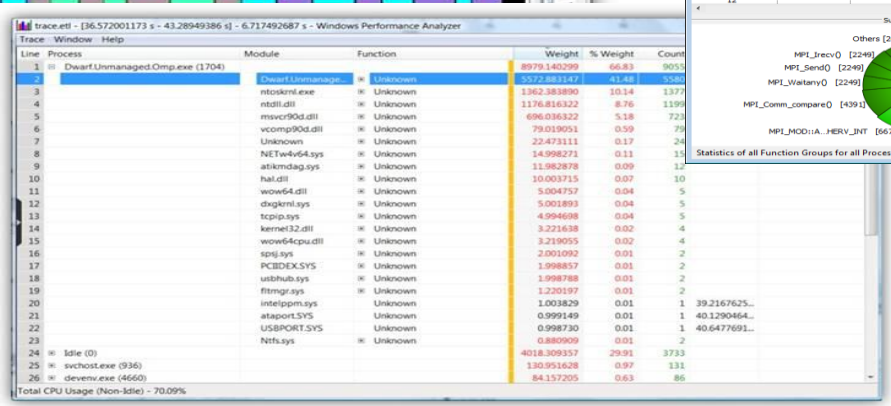




JumpShot: -PlotJumpshot -PlotVampir

Skip/Demo





Parallel Computing on Azure

- Demo:

Processing astronomical images for a TeraPixel panorama

The 10 Parsec Overview

4TB of raw images

- 2 X 1791 X 0.5 gigapixel plates
- Palomar (US) and UK Schmidt (Australia) telescopes
- Raw images have many artifacts
- Data access is challenging

Local + Cloud clusters

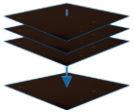
- Vignetting correction
- Astrometric alignment
- De-vignetting & color correction
- Stich & Smooth

Final rendered image

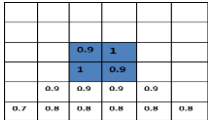
- 2^{40} (=1,099,511,627,776) pixels (RGB)
- Smooth, zoomable
- Silverlight or Desktop viewers
- Accessible by everyone

Color Plates Generation

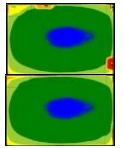
Vignetting correction



Creating Flat Fields



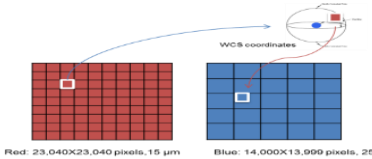
Normalization Matrix

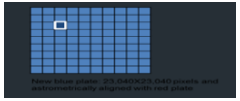


Normalizing Corners

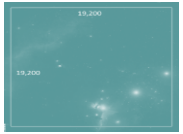
Devignetting

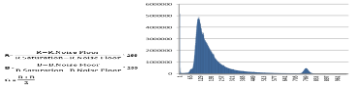
Astrometric Alignment

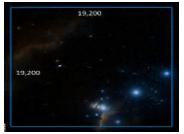




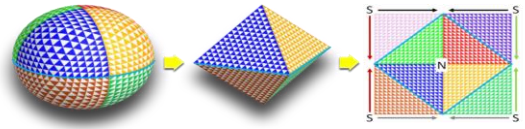
Color Correction



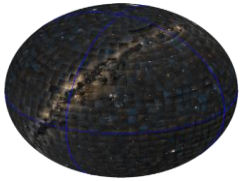


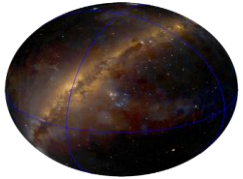


Stitching & Smoothing



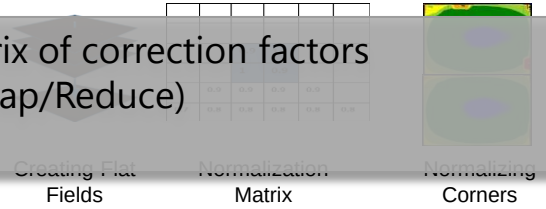
&





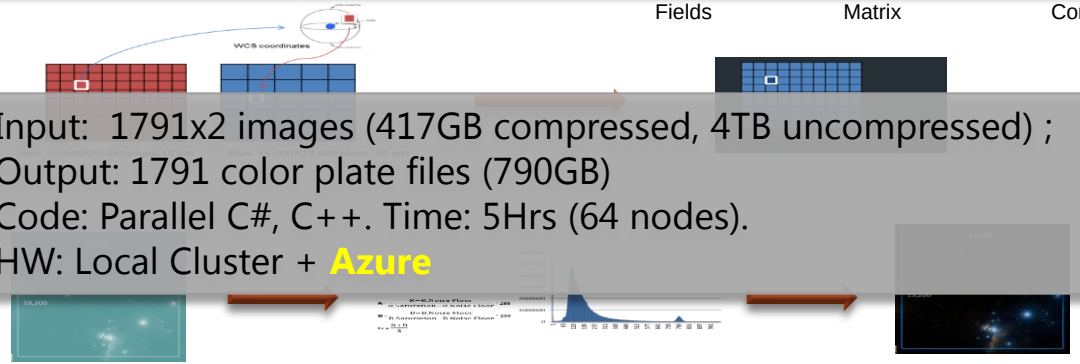
Vignetting Correction

- Input: 1791 X 2 images; Output: matrix of correction factors
- Code: Parallel C#, C++, DryadLinq (Map/Reduce)
- HW: Local Cluster



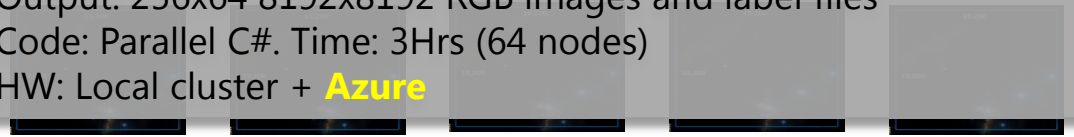
Color Plates Generation

- Input: 1791x2 images (417GB compressed, 4TB uncompressed) ; Output: 1791 color plate files (790GB)
- Code: Parallel C#, C++. Time: 5Hrs (64 nodes).
- HW: Local Cluster + **Azure**



Stitching

- Output: 256x64 8192x8192 RGB images and label files
- Code: Parallel C#. Time: 3Hrs (64 nodes)
- HW: Local cluster + **Azure**



Smoothing

- Final output: 1025 image pyramids (802G);
- Code: C++, MPI, Parallel C#; Time: Smooth=4Hrs; Zoom=40m
- HW: Local Cluster (64 nodes); 2.5 Hrs to move final data off cluster (1Gpbs)

- Would require 500,000 HDTVs to view
- Stretched out, would fill an American football field

Technical Details/Demo

- Acquisition & node mgmt on Azure
- Visual Studio : DryadLinq
- Visual Studio : .Net Parallel Extensions
- Visual Studio : MPI

The DryadLinq Query

```
DryadLinq.UsePLINQ = false;
DryadLinqTools.RemoveUnwantedDllsFromResourceSet();

var pixelRows = folders.SelectMany(image => ImageToRows(image, options));
var stackedPixelRows = pixelRows.GroupBy(pixelRow => pixelRow.Position);
var finalRows = stackedPixelRows.Select(x => ReduceStackedRows(x));
var b = finalRows.Apply(x => SaveFlatField(x, options));
return b.Single();
```

.Net Parallel Extensions

```
Parallel.For(0, plate.Height, (y) =>
        {
            //Flip the y-index because a DSS plate has y=0 at the bottom of
            //the image but a bitmap has the origin at the top
         int iy = plate.Height - 1 - y;
            int pos = y * imageData.Stride;
            for (int ix = 0; ix < plate.Width; ix++)
            {
                Color c = scale.Map(plate.Data[iy][ix]);
                rgb[pos++] = c.B;
                rgb[pos++] = c.G;
                rgb[pos++] = c.R;
            }
         });
```

What's actually nicer on windows for TC work
(in my humble opinion)

- Visual Studio, C#, F#, C++ IDE
- MPI.net
- Graphics & GPGPU drivers, Nvidia Nsight, …
- Python MPI
- CAS packages
- TC / Domain Specialist support: eg run Excel at scale
- HPC cluster setup / mgmt / multi-discipline usage
- Intel & PGI's compiler integration
- .Net in some ways > JVM

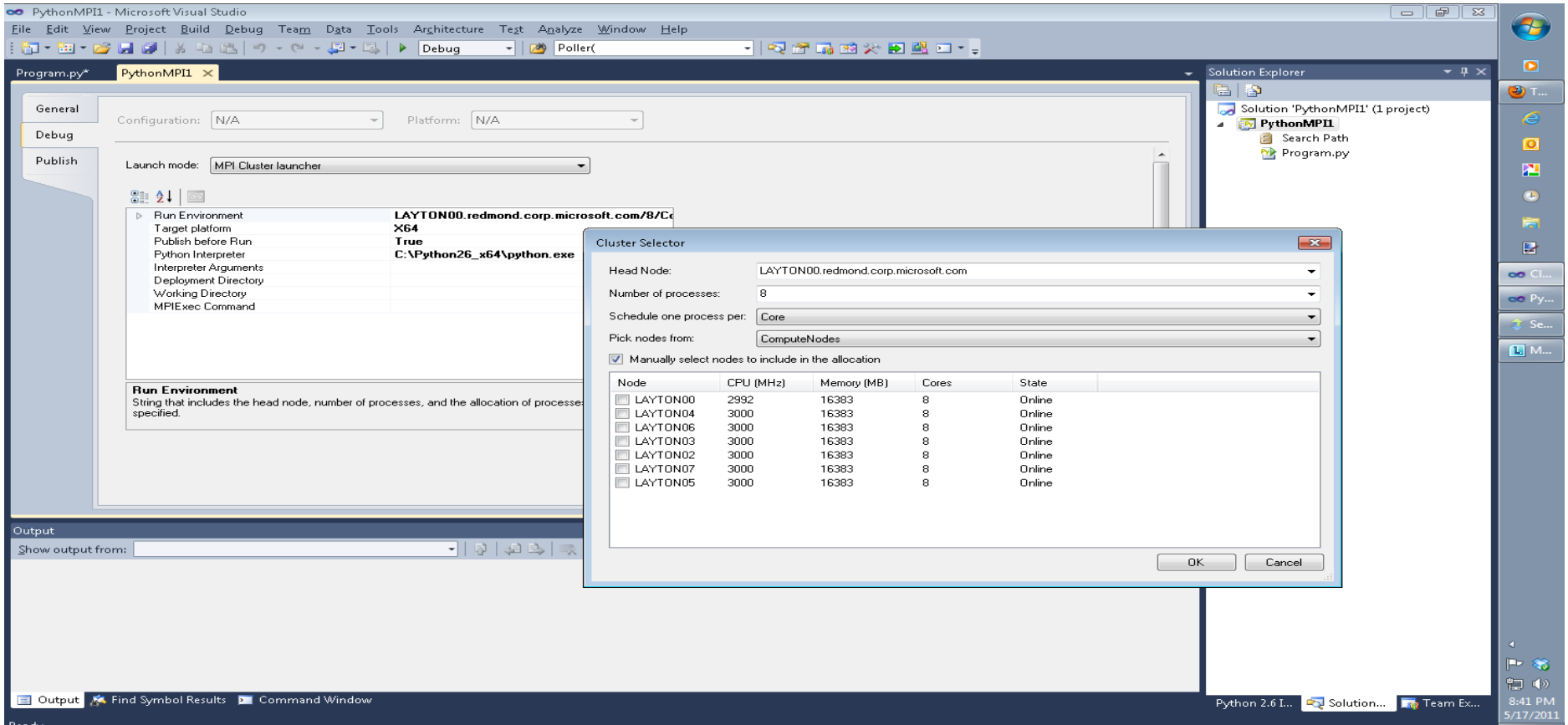
Conclusion…

- If you haven't already, give your Windows box some love!
- Leverage your investments: tools, code, muscle memory, …
- Cross-platform languages/runtimes enable kicking the tires without lock-in

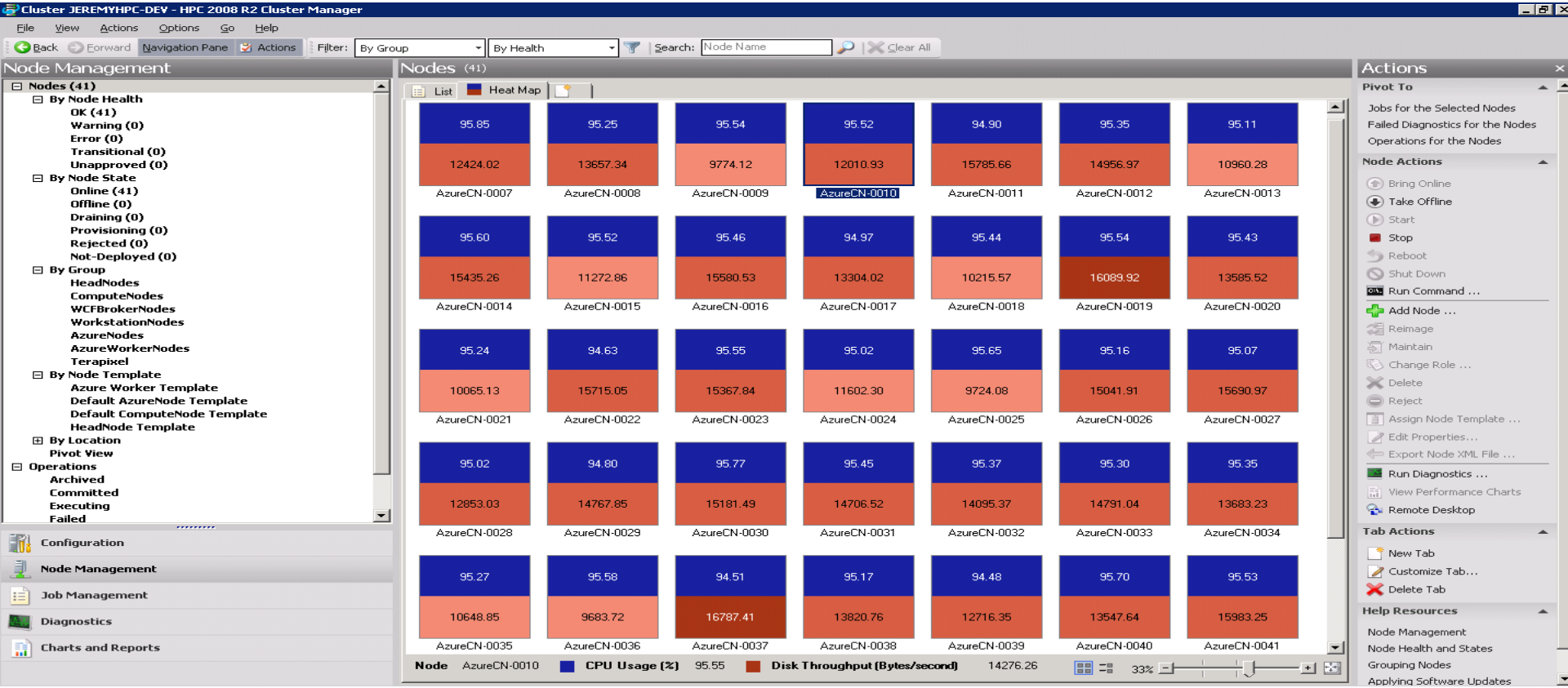
Winux™ : The best features of Linux & Windows !

Backup screen shots

Python Tools for Visual Studio
Free/OSS



Burst to Azure



Development

- Porting vs developing new code
- Examine headers, code, platform dependencies
- UI layer
- Support libraries
- make
- Compile/link/build, debug, iterate
- Profile, optimize
- Package & Deploy