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:
  😞

(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

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

- CAS environments

"The Unix look & feel"

General purpose development

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
  o Openoffice, Sun Java, ...
☞ License, who
  o GPL v2, redhat
☞ Useful links
  o www.cygwin.com
SUA
Subsystem for Unix Applications

- **What is it?**

- **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:**
MKS Toolkit

• **What is it?**

• **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](http://mkssoftware.com) general
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+ utilś – 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, ...

```c
# 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/mac os via Mono
Library/runtime Examples

- Boost
- Python: SciPy, Numpy, scikits, ...
- MKL
- IMSL
- NAG
- Visual Numerics
- LibFlame
- ATLAS
- MPI, OpenMP, CUDA, ...
- NetCDF, HDF5, FITS, ...

- From text to image processing to large graphs & MPI
- Numeric & Scientific libraries
- Highly optimized Math libraries for native & .Net number crunching
- Optimized parallel libraries for multicore, cluster and GPUs
- 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
Computer Algebra Systems

Commercial

Maple *
MATLAB *
Mathematica *

Free/OSS

Sage
IPython *
Octave

* 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](http://www.dreamspark.com) program
  • Free to startups via [www.bizspark.com](http://www.bizspark.com) program
• 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
• Python Tools for Visual Studio
  • Intellisense
  • Profiling
  • Cluster MPI debugging
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
Why Python

• Well suited to Technical & Scientific Computing
  • Isn’t it interpreted? (and slow?!)  
• Easy ramp up, yet powerful language
• Incredibly rich ecosystem of high quality libraries
• Healthy developer eco-system
• Various implementations
• Free, open source w quality distro’s providing support
  
  • Interactive by design
  • Easily mix in native code, even CUDA, MPI, ... see Cython, swig, ctypes
  • From CS101 to PhD thesis
  • Web, numerics, symbolics, Bio, astronomy, ...
  • 2D, 3D viz, .... See scipy, scikits
  • #3 most popular in some surveys
  • PyCon is the main conference
  • CPython, IronPython, Jython, PyPy, ...
  • BSD like license
  • Enthought Python Distro, ActivePython, ...
```
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 *namebuf = (char*)malloc(cnt * sizeof(char));
MPI_Gatherv(name, resultlen, MPI_CHAR, namebuf, 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] = namebuf + rdispls[i];
    qsort(hostnames, size, sizeof(char*), strcmp);
    for (int i = 0; i < size; ++i)
        printf("%s\n", hostnames[i]);
}
```

MPI.Net: A high performance wrapper for MPI

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

http://www.osl.iu.edu/research/mpi.net/
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
    }
}
NetPIPE Performance

Throughput (Mbps) vs. Message Size (Bytes)

- C (Native)
- C# (Primitive)
- C# (Serialized)
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

http://paralleldwarfs.codeplex.com
### Dwarf Popularity

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

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
StructuredGrid code fragment using MPI.NET
Results Summary: -PlotExcel
JumpShot: -PlotJumpshot -PlotVampir
Parallel Computing on Azure

• Demo:

  Processing astronomical images for a TeraPixel panorama
The 10 Parsec Overview

Collect
- 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

Process
- Local + Cloud clusters
  - Vignetting correction
  - Astrometric alignment
  - De-vignetting & color correction
  - Stich & Smooth

View
- Final rendered image
  - 2^{30} (=1,099,511,627,776) pixels (RGB)
  - Smooth, zoomable
  - Silverlight or Desktop viewers
  - Accessible by everyone
Creating Flat Fields
Normalizing Matrix
Normalizing Corners

Vignetting correction
Devignetting
Astrometric Alignment
Color Plates Generation
Color Correction
Stitching & Smoothing
Creating Flat Fields Normalization Matrix Normalizing Corners

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

- Collect
- View

Vignetting Correction

Color Plates Generation

Stitching

Smoothing

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

- 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

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

- 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
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();
```csharp
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
### Node Management

#### By Node Health
- Critical (1)
- Warning (0)
- Error (0)
- Transitional (0)
- Unapproved (0)

#### By Node State
- Online (41)
- Offline (6)
- Draining (0)
- Provisioning (0)
- Rejected (0)
- Not-Deployed (0)

#### By Group
- HeadNodes
- ComputeNodes
- WCNWorkerNodes
- WorkstationNodes
- AzureNodes
- AzureWorkerNodes
- Terapsed

#### By Node Template
- Azure Worker Template
- DefaultAzureNode Template
- Default Compute Node Template
- HeadNode Template

#### By Location
- Per View

#### Operations
- Archived
- Commited
- Executing
- Failed

---

### Nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>CPU Usage (%)</th>
<th>Disk Throughput (Bytes/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AzureCN-0001</td>
<td>55.15</td>
<td>1034.42</td>
</tr>
<tr>
<td>AzureCN-0002</td>
<td>55.52</td>
<td>1036.94</td>
</tr>
<tr>
<td>AzureCN-0003</td>
<td>55.54</td>
<td>1075.52</td>
</tr>
<tr>
<td>AzureCN-0004</td>
<td>55.55</td>
<td>1078.34</td>
</tr>
<tr>
<td>AzureCN-0005</td>
<td>55.56</td>
<td>1080.10</td>
</tr>
<tr>
<td>AzureCN-0006</td>
<td>55.57</td>
<td>1082.96</td>
</tr>
<tr>
<td>AzureCN-0007</td>
<td>55.58</td>
<td>1085.82</td>
</tr>
<tr>
<td>AzureCN-0008</td>
<td>55.59</td>
<td>1088.70</td>
</tr>
<tr>
<td>AzureCN-0009</td>
<td>55.60</td>
<td>1091.60</td>
</tr>
<tr>
<td>AzureCN-0010</td>
<td>55.61</td>
<td>1094.50</td>
</tr>
<tr>
<td>AzureCN-0011</td>
<td>55.62</td>
<td>1097.40</td>
</tr>
<tr>
<td>AzureCN-0012</td>
<td>55.63</td>
<td>1099.30</td>
</tr>
<tr>
<td>AzureCN-0013</td>
<td>55.64</td>
<td>1102.20</td>
</tr>
<tr>
<td>AzureCN-0014</td>
<td>55.65</td>
<td>1105.10</td>
</tr>
<tr>
<td>AzureCN-0015</td>
<td>55.66</td>
<td>1108.00</td>
</tr>
<tr>
<td>AzureCN-0016</td>
<td>55.67</td>
<td>1110.90</td>
</tr>
<tr>
<td>AzureCN-0017</td>
<td>55.68</td>
<td>1113.80</td>
</tr>
<tr>
<td>AzureCN-0018</td>
<td>55.69</td>
<td>1115.70</td>
</tr>
<tr>
<td>AzureCN-0019</td>
<td>55.70</td>
<td>1117.60</td>
</tr>
<tr>
<td>AzureCN-0020</td>
<td>55.71</td>
<td>1119.50</td>
</tr>
<tr>
<td>AzureCN-0021</td>
<td>55.72</td>
<td>1121.40</td>
</tr>
<tr>
<td>AzureCN-0022</td>
<td>55.73</td>
<td>1123.30</td>
</tr>
<tr>
<td>AzureCN-0023</td>
<td>55.74</td>
<td>1125.20</td>
</tr>
<tr>
<td>AzureCN-0024</td>
<td>55.75</td>
<td>1127.10</td>
</tr>
<tr>
<td>AzureCN-0025</td>
<td>55.76</td>
<td>1129.00</td>
</tr>
<tr>
<td>AzureCN-0026</td>
<td>55.77</td>
<td>1130.90</td>
</tr>
<tr>
<td>AzureCN-0027</td>
<td>55.78</td>
<td>1132.80</td>
</tr>
<tr>
<td>AzureCN-0028</td>
<td>55.79</td>
<td>1134.70</td>
</tr>
<tr>
<td>AzureCN-0029</td>
<td>55.80</td>
<td>1136.60</td>
</tr>
<tr>
<td>AzureCN-0030</td>
<td>55.81</td>
<td>1138.50</td>
</tr>
<tr>
<td>AzureCN-0031</td>
<td>55.82</td>
<td>1140.40</td>
</tr>
<tr>
<td>AzureCN-0032</td>
<td>55.83</td>
<td>1142.30</td>
</tr>
<tr>
<td>AzureCN-0033</td>
<td>55.84</td>
<td>1144.20</td>
</tr>
<tr>
<td>AzureCN-0034</td>
<td>55.85</td>
<td>1146.10</td>
</tr>
<tr>
<td>AzureCN-0035</td>
<td>55.86</td>
<td>1148.00</td>
</tr>
<tr>
<td>AzureCN-0036</td>
<td>55.87</td>
<td>1150.00</td>
</tr>
<tr>
<td>AzureCN-0037</td>
<td>55.88</td>
<td>1152.00</td>
</tr>
<tr>
<td>AzureCN-0038</td>
<td>55.89</td>
<td>1154.00</td>
</tr>
<tr>
<td>AzureCN-0039</td>
<td>55.90</td>
<td>1156.00</td>
</tr>
<tr>
<td>AzureCN-0040</td>
<td>55.91</td>
<td>1158.00</td>
</tr>
<tr>
<td>AzureCN-0041</td>
<td>55.92</td>
<td>1160.00</td>
</tr>
</tbody>
</table>

---

**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