Lecture 11
ECEN 5653
Code Configuration Management and Version Control, User-Space Debug And Performance Optimizations
Overview

- NAB Show This Week - http://www.nabshow.com/
- Viral Video – The Bobs (Consultants)
- Extended Lab Discussion
  - Too Easy, Too Hard?
  - Questions?
- Tips For Extended Lab
  - Code Management with SVN
  - Debugging with DDD
  - Getting NVIDIA CUDA 3.0 Working on Your Box!
  - GCC –O0, -O1, -O2, -O3
  - Disassembly
  - SSE code generation
Easy SVN

yum install svn
yum install svnserv

svnadmin create /usr/local/svn/testrepos

vi /usr/local/svn/testrepos/conf/svnserv.conf
[general]
### These options control access to the repository for unauthenticated
### and authenticated users. Valid values are "write", "read",
### and "none". The sample settings below are the defaults.
anon-access = read
anon-access = write

svnserv -d -r /usr/local/svn
Easy SVN Continued

svn checkout svn://localhost/testrepos testrepos

[r0ot@localhost testrepos]# cp -r ../ssd .
[root@localhost testrepos]# ls
ssd
[root@localhost testrepos]# svn add ssd
A      ssd
A      ssd/Makefile
A      ssd/ssd_unload
A      ssd/ssd.c
A      ssd/Module.symvers
A      ssd/testssd.c
A      ssd/ssd.h
A      ssd/ssd_load
[root@localhost testrepos]# svn commit -m "testing"
Adding  ssd
Adding  ssd/Makefile
Adding  ssd/Module.symvers
Adding  ssd/ssd.c
Adding  ssd/ssd.h
Adding  ssd/ssd_load
Adding  ssd/ssd_unload
Adding  ssd/testssd.c
Transmitting file data .......
Committed revision 1.
[root@localhost testrepos]#
More Advanced SVN

Read the Book
- http://svnbook.red-bean.com/

Backups Not Provided by SVN
- Backup your repository, esp. if you use one on the lab Linux boxes
  - As simple as tarball of /usr/loca/svn/myrepository

Carefully Consider Options for Repository Access – URL
- Example Give for svnsserve
  - Can use SSH tunneling
  - Can use HTTPS (Secure http)
  - Many other options…
Linux Debugging

- **DDD – Single Step Debugging**
  - Yum install DDD
  - DDD executable
  - Demo with Linux

- **Single Step Debugging Does Not Help with Timing Issues or Execution Efficiency**
  - Race Conditions
  - Missing Deadlines (Deadlock, Unbounded Prio Inversion, Inefficient Code)

- **LTT Next Generation and PMU Tools Can Help with Timing and Efficiency**
  - [http://ltt.polymtl.ca/](http://ltt.polymtl.ca/)
  - [Vtune for Linux](http://www.in Alto.com/products/vtune)
Profiling Code and System

- **Sysprof** – simplest way to profile your application and all other system services (daemons) and their CPU use
  - http://sysprof.com/

- **Compiler-based Profiling Tool**
  - $ gcc -o prog -g -p prog.c
  - $ ./prog
  - $ gprof -b ./prog

- **Qprof**

- **/proc** – pseudo filesystem to access kernel values
  - /proc/stat, /proc/meminfo, …

- **top** – provides a snapshot of process information
Sysprof Example for Color Transformation on 1080p
Scaling Post Production

CPU & IO Threading, Scaling, and Speed-up
Parallel Processing Speed-up

- Grid Data Processing Speed-up
  1. Multi-Core, Multi-threaded, Macro-blocks/Frames
  2. SIMD, Vector Instructions Operating over Large Words (Many Times Instruction Set Size)
  3. Co-Processor Operates in Parallel to CPU(s)

- SPMD – GPU or GP-GPU Co-Processor
  - PCI-Express Bus Interfaces
  - Transfer Program and Data to Co-Processor
  - Threads and Blocks to Transform Data Concurrently

- Image Data Processing – Few Data Dependencies
  - Good Speed-up by Amdahl’s Law
  - $P=\text{Parallel Portion}$
  - $(1-P)=\text{Sequential Portion}$
  - $S=\#\text{ of Cores (Concurrency}$)
  - Overhead for Co-Processor
  - IO for Co-Processing

\[
\text{Max Speed Up} = \frac{1}{(1-P)+0}
\]

\[
\text{Multicore Speed Up} = \frac{1}{(1-P)+\frac{P}{S}}
\]
Amdahl’s Law – Infinite Cores

- Maximum Speed-up Driven by Sequential and Parallel Portions of Program
  - $P =$ Parallel Portion
  - $(1-P) =$ Sequential Portion
  - Speed-up for Given Multi-core Architecture Function of # of Cores (Speed-up in Parallel Portions)

Amdahl's Law Max Speed-up
(Any Number of Processor Cores)

Algorithm Speed Up

Sequential Portion (% Computation in Sequential vs. Parallel Execution)
Multi-Core Speed-Up

Amdahl's Law - Speed-up with # Cores and Parallel Portion

- Max Speed-up
- 2 cores
- 4 cores
- 8 cores
- 12 cores
- 32 cores

95% Parallel Program

© Sam Siewert
Hiding IO Latency – Overlapping with Processing

Simple Design – Each Thread has READ, PROCESS, WRITE-BACK Execution

Frame rate is READ+PROCESS+WRITE latency – e.g. 10 fps for 100 milliseconds

- If READ is 70 msec, PROCESS is 10 msec, and WRITE-BACK 20 msec, predominate time is IO time, not processing
- Disk drive with 100 MB/sec READ rate can only read 16 fps, 62.5 msec READ latency
Hiding IO Latency

Schedule Multiple Overlapping Threads?

- Requires $N_{\text{threads}} = N_{\text{stages}} \times N_{\text{cores}}$
- 1.5 to 2x Number of Threads for SMT (Hyper-threading)
- For IO Stage Duration Similar to Processing Time
- More Threads if IO Time (Read+WB+Read) $>>$ 3 x Processing Time

© Sam Siewert
Hiding Latency – Dedicated IO

Schedule Reads Ahead of Processing

- Requires \( N_{\text{threads}} = 2 + N_{\text{cores}} \)
- Synchronize Frame Ready/Write-backs
- Balance Stage Read/Write-Back Latency to Processing
- 1.5 to 2x Threads for SMT (Hyper-threading)
Processing Latency Alone

- Write Code with Memory Resident Frames
  - Load Frames in Advance
  - Process In-Memory Frames Over and Over
  - Do No IO During Processing
  - Provides Baseline Measurement of Processing Latency per Frame Alone
  - Provides Method of Optimizing Processing Without IO Latency
IO Latency Alone

- Comment Out Frame Transformation Code or Call Stubbed NULL Function
  - Provides Measurement of IO Frame Rate Alone
  - Essentially Zero Latency Transform
  - No Change Between Input Frames and Output Frames
  - Allows for Tuning of IO Scheduler and Threading

© Sam Siewert
Tips for IO Scheduling

- `blockdev --getra /dev/sda`
  - Should return 256
  - Means that reads read-ahead up to 128K
  - Function calls – read, fread should request as much as possible
  - Check “actual bytes read”, re-read as needed in a loop

- `blockdev --setra /dev/sda 16384 (8MB)`

- **Switch CFQ to Deadline**
  - Use “lsscsi” to verify your disk is /dev/sda … substitute block driver interface used for file system if not sda
  - `cat /sys/block/sda/queue/scheduler`
  - `echo deadline > /sys/block/sda/queue/scheduler`

- Options are noop, cfq, deadline
NVIDIA CUDA 3.0

Installation and Test
Simple Benchmarks
More CUDA Installation Hints

- Note that Linux Currently DOES NOT WORK with any Optimus Hybrid Power Management NVIDIA cards (beware, common on laptops)

- Install Your GeForce, Tesla, etc. and Verify on PCI Bus with lspci
  - Use update-pciids if you don’t see your NVIDIA card
  - lspci | grep –i nvidia

- First Install the standard NVIDIA driver and make sure your X-Windows work with it

- Verify NVIDIA settings with /usr/bin/nvidia-settings

- Download Developer Driver (to replace driver above) , CUDA toolkit, and SDK

- Download “Getting Started”

- Install Driver, and If you See an ERROR
  - Exit X with /sbin/init 3
  - Use “sh driver-install-blah-blah.run”
  - Check /var/log/nvidia-installer.log
  - Use init 5 or startx to re-run X windows

- If you have a driver Module Load Issue On FC, Blacklist Current X11 driver known as “nouveau”, but if you followed steps above should not be an issue
  - See http://fedorasolved.org/video-solutions/nvidia-yum-kmod
  - Set up RPM Fusion Repository for YUM
  - Check Kernel Revision/Type with “uname –r”
  - Follow Detailed Instructions on Blacklisting nouveau for X11
  - Reboot
  - X Windows Should Show NVIDIA Splash – If not, See Troubleshooting on same page, go back to “Verify NVIDIA settings”

- Re-install NVIDIA driver if needed, Check with:
  - /usr/bin/nvidia-settings – should come up without any errors
  - X-windows Should run on your GeForce

- After installing Developer Driver, CUDA toolkit, download and build examples and run deviceQuery and bandwidthTest to
Building Examples

- Follow Downloaded “Getting Started”
- Install CUDA toolkit
- Yum install freeglut-devel so you can run openGL
- Install GPU SDK
- Set $PATH and $LD_LIBRARY_PATH
- Now Try NVIDIA_GPU_Computing_SDK/C and “make”
- Turn SELinux to DISABLED, PERMISSIVE
- If you have issues with some builds on FC14 and CUDA 3.2, just skip them – as long as you can run deviceQuery and bandwidthTest, you are ok
Play with CUDA Examples

C/bin/linux/release - ./deviceQuery

./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

There is 1 device supporting CUDA

Device 0: "GeForce 8700M GT"
CUDA Driver Version: 3.0
CUDA Runtime Version: 3.0
CUDA Capability Major revision number: 1
CUDA Capability Minor revision number: 1
Total amount of global memory: 536150016 bytes
Number of multiprocessors: 4
Number of cores: 32
Total amount of constant memory: 65536 bytes
Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 8192
Warp size: 32
Maximum number of threads per block: 512
Maximum sizes of each dimension of a block: 512 x 512 x 64
Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
Maximum memory pitch: 2147483647 bytes
Texture alignment: 256 bytes
Clock rate: 1.25 GHz
Concurrent copy and execution: Yes
Run time limit on kernels: Yes
Integrated: No
Support host page-locked memory mapping: No
Compute mode: Default (multiple host threads can use this device simultaneously)

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 134562363, CUDA Runtime Version = 3.0, NumDevs = 1, Device = GeForce 8700M GT

PASSED

© Sam Siewert
./bandwidthTest Starting...

Running on...

Device 0: GeForce 8700M GT
Quick Mode

<table>
<thead>
<tr>
<th>Transfer Size (Bytes)</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33554432</td>
<td>1645.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transfer Size (Bytes)</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33554432</td>
<td>1123.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transfer Size (Bytes)</th>
<th>Bandwidth (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33554432</td>
<td>15037.5</td>
</tr>
</tbody>
</table>

[bandwidthTest] - Test results: PASSED
N-body, etc. …

Test Environment, Measure Performance
Hands On Session and Practice

- Q&A on Prof’s Virtual Box with Systemtap and Sysprof
- Read Programmer’s Guide Next and Browse Example Source
- Read “Benchmarking GPU Devices with N-Body Simulations” – On our Web site
- Study SSE Intel Paper Example
  - Disassemble Optimized Code and Step Through in Mixed Mode
  - Look at Code Generated for –O0, -O1, -O2, -O3
CUDA, OpenCL, StreamProc

CUDA is NVIDIA Driver for GPU and GP-GPU Interface
- Host – Device memory and DMA
- Kernel download and execution on device

OpenCL is a Portable Alternative (Intel, AMD, NVIDIA)
- Intel OpenCL for SSE and Atom-based Multi-core
- AMD ATI OpenCL for StreamProc
- OpenCL Driver for CUDA

With CUDA or OpenCL, key is Kernel Design for Gridded Data Transformation and Efficient Host-Device Data Transfer and Overlap with Sequential CPU code

References:
  - http://www.elsevierdirect.com/morgan_kaufmann/kirk/
- “OpenCL Programming Guide”, Aaftab Munshi
Thank You! – Hoshi & Kaibu