The PARSEC Benchmark Suite

PARSEC 3.0

ARCO Group seminar
27th January of 2012

Martí Torrents Lapuerta
Outline

- Understanding PARSEC
  - Introduction, History, What’s New
  - Workloads

- Working with PARSEC
  - The parsecmgmt tool
  - Building & Running workloads
  - Configuration files

- Concluding remarks
Outline

- Understanding PARSEC
  - Introduction, History, What’s New
  - Workloads

- Working with PARSEC
  - The parsecmgmt tool
  - Building & Running workloads
  - Configuration files

- Concluding remarks
What is PARSEC

- Princeton Application Repository for Shared-Memory Computers
- Benchmark Suite for Chip-Multiprocessors
- Started as a cooperation between Intel and Princeton University, many more have contributed since then
- Freely available at:
  - http://parsec.cs.princeton.edu/
- Anybody can use it for its research
Introduction to PARSEC

• **Goal:**
  – An open-source parallel benchmark suite of emerging applications for evaluating multi-core and multiprocessor systems

• **Application domains:**
  – financial, computer vision, physical modeling, future media, content-based search, deduplication

• **Current release:**
  – PARSEC 2.1 (13 applications)
History of PARSEC

• Jan 2008 PARSEC 1.0
  – 12 workloads

• Feb 2009 PARSEC 2.0
  – One new workload, raytrace

• Aug 2009 PARSEC 2.1
  – Bugfix

• **PARSEC 3.0**
  – Summer 2011
History of PARSEC

• Jan 2008 PARSEC 1.0
  – 12 workloads

• Feb 2009 PARSEC 2.0
  – One new workload, raytrace

• Aug 2009 PARSEC 2.1
  – Bugfix

• PARSEC 3.0
  – Summer 2011

So When?
Features of new PARSEC 3.0

• New framework
  – Support network workloads
  – Be more convenient to add new workloads

• Improved workloads
  – blackscholes, bodytrack, canneal, dedup, facesim, ferret, fluidanimate, freqmine, vips.

• Not modified workloads
  – raytrace, streamcluster, swaptions, x264

• SPLASH-2
  – Existing SPLASH-2 using the same framework
    – Use parsecmgmt to manage, build, and run
Workloads (I/IV)

1. Blackscholes: Simplest of all PARSEC workload
   - Small working sets, negligible communication

2. Bodytrack: Computer vision workload
   - Medium working sets, some communication

3. Canneal: Workload with most demanding memory behavior
   - Huge working sets, busty communication

4. Dedup: Data stream compression (pipelined model)
   - Huge working sets, significant communication
Workloads (II/IV)

5. Facesim: Physical faces simulation
   Large working sets, some sharing

6. Ferret: Content-based similarity search (similar to Goggles)
   Huge working sets, very communication intensive

7. Fluidanimate: Smoke and particles effects in animations
   Large working sets, some communication

8. Freqmine: Identifies patterns in a transaction database
   Huge working sets, some sharing
Workloads (III/IV)

9. Raytrace: Real-time animation for computer games
   Large working sets, little communication

10. Streamcluster: It assigns each point of a stream to its nearest center
    Medium-sized working sets of user-determined size

11. Swaptions: Applies some simulations to some options
    Medium-sized working sets, little communication

12. Vips: Image transformation
    Medium-sized working sets, some sharing
13. X264: Video encoding (MPEG-4)
Medium working sets, very communication intensive
## Workloads (Summary)

<table>
<thead>
<tr>
<th>Program</th>
<th>Application Domain</th>
<th>Parallelization Model</th>
<th>Granularity</th>
<th>Working Set</th>
<th>Data Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>Financial Analysis</td>
<td>data-parallel</td>
<td>coarse</td>
<td>small</td>
<td>low</td>
</tr>
<tr>
<td>bodytrack</td>
<td>Computer Vision</td>
<td>data-parallel</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>canneal</td>
<td>Engineering</td>
<td>unstructured</td>
<td>fine</td>
<td>unbounded</td>
<td>high</td>
</tr>
<tr>
<td>dedup</td>
<td>Enterprise Storage</td>
<td>pipeline</td>
<td>medium</td>
<td>unbounded</td>
<td>high</td>
</tr>
<tr>
<td>facesim</td>
<td>Animation</td>
<td>data-parallel</td>
<td>coarse</td>
<td>large</td>
<td>low</td>
</tr>
<tr>
<td>ferret</td>
<td>Similarity Search</td>
<td>pipeline</td>
<td>medium</td>
<td>unbounded</td>
<td>high</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>Animation</td>
<td>data-parallel</td>
<td>fine</td>
<td>large</td>
<td>low</td>
</tr>
<tr>
<td>freqmine</td>
<td>Data Mining</td>
<td>data-parallel</td>
<td>medium</td>
<td>unbounded</td>
<td>high</td>
</tr>
<tr>
<td>raytrace</td>
<td>Rendering</td>
<td>data-parallel</td>
<td>medium</td>
<td>unbounded</td>
<td>high</td>
</tr>
<tr>
<td>streamcluster</td>
<td>Data Mining</td>
<td>data-parallel</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>swaptions</td>
<td>Financial Analysis</td>
<td>data-parallel</td>
<td>coarse</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>vips</td>
<td>Media Processing</td>
<td>data-parallel</td>
<td>coarse</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>x264</td>
<td>Media Processing</td>
<td>pipeline</td>
<td>coarse</td>
<td>medium</td>
<td>high</td>
</tr>
</tbody>
</table>
Outline

Understanding PARSEC
- Introduction, History, What’s New
- Workloads

Working with PARSEC
- The parsecmgmt tool
- Building & Running workloads
- Configuration files

Concluding remarks
Framework Directory structure

- Framework executable
- Global Configuration files
- Extended benchmark directory
- PARSEC benchmark directory
- Each group directory contains one directory per package in that group
Framework Directory structure

[PACKAGENAME]/

inputs/  →  Input archives (optional)

inst/  →  Build installations with one subdirectory per installation

...  

obj/  →  Build directory for temporary files, one subdirectory per build

...  

parsec/  →  Local configuration files

run/  →  Run for temporary files

src/  →  Source code package

...
Environment Setup

• You can modify your environment to make the PARSEC tools and its man pages available at the command line (without full path)

• The `env.sh` script in the PARSEC root directory will do that for you

• Source it as follows (example assumes bash shell):

  ```
  source env.sh
  ```

• If you use PARSEC a lot you can add that to your login scripts to have it always available
Parsecmgmt command

- A script to help you manage your PARSEC installation
- Can build and run PARSEC workloads for you
- Only there for convenience, you can also do the same tasks manually
- Uses information in configuration files to do its job
- Use the following command to get some help:

  ```bash
  parsecmgmt -h
  ```
Build with parsecmgmt

- You can build a PARSEC workload as follows:
  ```
  parsecmgmt -a build -p [suite].[PACKAGE]
  ```
- Flag `-a` specifies the desired action, flag `-p` gives one or more packages
- A package can be a workload, library or anything else that comes with PARSEC and can be compiled
- `parsecmgmt -a info` gives you a list of all available packages
- `parsecmgmt` will automatically handle dependencies between packages correctly
Build with parsecgmgt

- Ex1: How do you build workload canneal?
- Ex2: How do you build workload raytrace in parsec suite?
- Ex3: How do you build workload raytrace in splash2x suite?
Build with parsecmgmt

- Ex1: How do you build workload canneal?
- With the following command:

```bash
> parsecmgmt -a build -p canneal
[PARSEC] Packages to build: canneal
[PARSEC] [========== Building package canneal==========]
[PARSEC] [----------Analyzing package canneal----------]
[PARSEC] canneal depends on: hooks
[PARSEC] [----------Analyzing package hooks----------]
[PARSEC] hooks does not depend on any other packages.
[PARSEC] [----------Building package hooks----------]
[PARSEC] Copying source code of package hooks.
[PARSEC] Running 'envmake':
/usr/bin/gcc-03 -funroll-loops -fprefetch-loop-arrays
-DPARSEC_VERSION=2.0 -Wall -std=c99 -D_GNU_SOURCE
-D_XOPEN_SOURCE=600 -c hooks.c
arrcslibhooks.ahooks.o
ranliblibhooks.a
[PARSEC] Running 'envmake install':
...
```
Build with parsecmgmt

• Ex2: How do you build workload raytrace in parsec suite?
• Ex3: How do you build workload raytrace in splash2x suite?
• With the following commands:

  parsecmgmt -a build -p parsec.raytrace

  parsecmgmt -a build -p splash2x.raytrace
Suite, Groups & Aliases

- Each package belongs to exactly one group
- `parsecmgmt` also understands aliases
- You can use group names and aliases instead of package names
- Example:
  ```
  parsecmgmt-a build -p parsec
  parsecmgmt-a build -p all
  parsecmgmt-a build -p splash2x
  ```
- Current Suites are `parsec`, `splash2x`
- Possible aliases are `kernels`, `apps`, `bench`, `libs`, `tools` and `all`
Build Configurations

• Build configurations determine how `parsecmgmt` is to build a package
• Specifies compiler, compiler flags, optimizations, etc.
• Use flag `-c` with `parsecmgmt` to select a build configuration
• You should create your own build configurations according to your needs
• Default build configurations are `gcc`, `gcc-hooks`, `gcc-serial` and `icc`
• PARSEC build configurations to enable specific parallelizations are `gcc-openmp`, `gcc-pthreads` and `gcc-tbb`
Build Configurations

- **Ex1**: How do you build workload `canneal` with build configuration `gcc-serial`?

- **With the following command:**
  ```
  > parsecmgmt -a build -p canneal -c gcc-serial
  [PARSEC] Packages to build: canneal
  [PARSEC] [========== Building package canneal =========]
  [PARSEC] [----------Analyzing package canneal ----------]
  [PARSEC] canneal depends on: hooks
  [PARSEC] [----------Analyzing package hooks ----------]
  [PARSEC] hooks does not depend on any other packages.
  [PARSEC] [----------Building package hooks ----------]
  [PARSEC] Copying source code of package hooks.
  [PARSEC] Running 'env make':
  /usr/bin/gcc -O3 -funroll-loops -fprefetch-loop-arrays
  -DPARSEC_VERSION=2.0 -Wall -std=c99 -D_GNU_SOURCE
  -D_XOPEN_SOURCE=600 -c hooks.c
  ar rcs libhooks.a hooks.o
  ranlib libhooks.a
  [PARSEC] Running 'env make install':
  ```
Multiple Builds

- You can have more than one build of every package installed
- Parsecmgmt will create a platform description string to distinguish builds as follows:
  
  \texttt{[ARCHITECTURE]-[OSNAME].[BUILDCONF]}

- You can override this string by defining environment variable \texttt{PARSECPLAT}
- PARSEC 2.0 also allows you to append an extension to further distinguish builds
Show All Builds

- You can see a list of all installed builds if you run:
  
  `parsecmgmt -a status -p all`

- Parsecmgmt will list the platform description strings of all installed builds for each workload:

  [PARSEC] Installation status of selected packages:
  [PARSEC] blackscholes:
  [PARSEC] -no installations-
  [PARSEC] bodytrack:
  [PARSEC] -no installations-
  ...

  [PARSEC] canneal:
  [PARSEC] x86_64-linux-gnu.gcc
  [PARSEC] x86_64-linux-gnu.gcc-serial
  ...

Clean up

• Remove all temporary directories (used e.g. for building):

  parsecmgmt -a fullclean -p all

• Uninstall a specific installation:

  parsecmgmt -a uninstall -p [PACKAGE] -c [BUILDCONF]

• Uninstall everything:

  parsecmgmt -a fulluninstall -p all
Inputs sets

- Test
  - Execute program, as small as possible, best-effort execution path as real inputs
- Simdev
  - Stresses all machine parts required by larger input sets, same execution path as real inputs
- Simsmall
  - Like real inputs, runtime ~1s
- Simmedium
  - Like real inputs, runtime ~5s
- Simlarge
  - Like real inputs, runtime ~15s
- Native
  - Like real inputs, runtime ~15min
Run Benchmarks

• PARSEC benchmark can be executed as follows:

```
```

• Like building workloads, but you can also specify an input and the number of threads

Flag '-n' specifies the minimum number of threads. The actual number can be higher. You must use other techniques to limit the number of CPUs

• Default inputs are test, simdev, simsmall, simmedium, simlarge and native
Run Benchmarks

- **Ex1:** How do you run the serial version of workload canneal with input simsmall?
- **With the following command:**

```
> parsecmgmt -a run -p canneal -c gcc-serial -i simsmall

[PARSEC] Benchmarks to run: canneal
[PARSEC] [========== Running benchmark canneal ==========]
[PARSEC] Setting up run directory.
[PARSEC] Unpacking benchmark input 'simsmall'.
100000.nets
[PARSEC] Running '...':
[PARSEC] [----------Beginning of output----------]
PARSEC Benchmark Suite Version 2.0
Threadcount: 1
10000 moves per thread
Start temperature: 2000
...
[PARSEC] [----------End of output----------]
[PARSEC] Done.
```
Outline

Understanding PARSEC
   Introduction, History, What’s New
   Workloads

Working with PARSEC
   The parsecmgmt tool
   Building & Running workloads
   Configuration files

Concluding remarks
Concluding remarks

- It covers many different program behaviors
- Its benchmarks uses many CMPs resources
- Known by almost everybody (working in CMPs)
- Easy to use
- Many information available in Internet
- Few benchmarks (new benchmarks in PARSEC 3.0)
  - GPU benchmarks could be useful
  - Network benchmarks
References

• Yungang Bao, Christian Bienia, Kai Li. The PARSEC Benchmark Suite Tutorial Seminar Slides, ISCA, June 2011

• http://parsec.cs.princeton.edu/

• http://wiki.cs.princeton.edu/

Thanks for your attention

QUESTIONS
The PARSEC Benchmark Suite

PARSEC 3.0

ARCO Group seminar
27th January of 2012

Martí Torrents Lapuerta