The PARSEC Benchmark Suite Tutorial
- PARSEC 2.0 -

by

Christian Bienia, Princeton University
and
Kai Li, Princeton University

Tutorial Contents

- Part 1: Understanding PARSEC
  - Overview
  - PARSEC vs. SPLASH-2
  - Workloads
- Part 2: Working with PARSEC
  - The `parsecmgmt` tool
  - Building & Running workloads
- Part 3: Adapting PARSEC
  - Configuration files
  - PARSEC Hooks
- Part 4: Concluding Remarks

Part 1

Understanding PARSEC

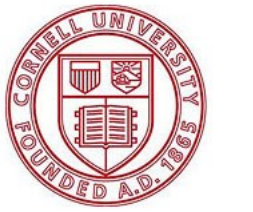
What is PARSEC?

- 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:

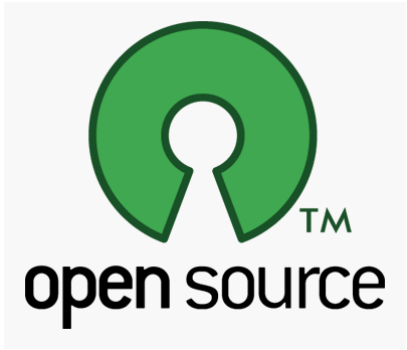
http://parsec.cs.princeton.edu/

- You can use it for your research

Contributors



Cornell University



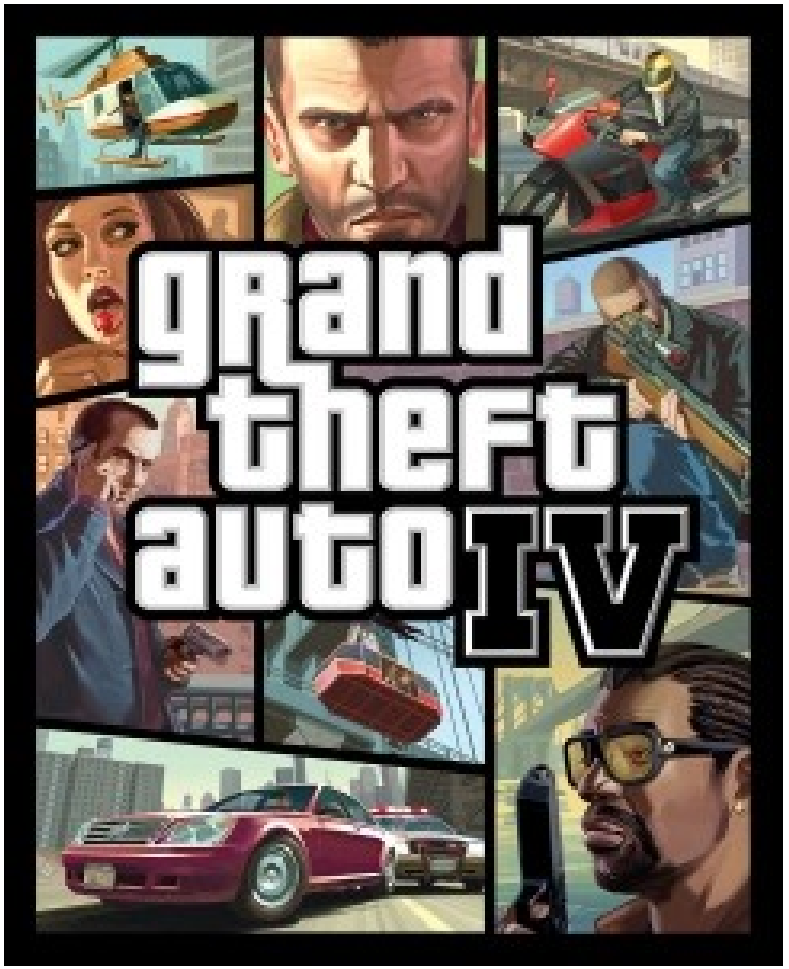
The first version of PARSEC was created by Intel and Princeton University.

We would like PARSEC to be a community project.

Many people and institutions have already contributed.

Why a new benchmark suite?

- Trend 1: New application areas for parallel machines through proliferation of CMPs
- Trend 2: Drastic change of architecture constraints driven by CMPs
- Trend 3: Explosion of globally stored data requires new algorithms



Rockstar's "Grand Theft Auto IV" (2008) requires CMPs

We need to consider new technology trends

Interest in PARSEC

Over 1,000 downloads from all over the world since release

Objectives of PARSEC

- Multithreaded Applications
  Future programs must run on multiprocessors
- Emerging Workloads
  Increasing CPU performance enables new applications
- Diverse
  Multiprocessors are being used for more and more tasks
- State-of-Art Techniques
  Algorithms and programming techniques evolve rapidly
- Support Research
  Our goal is insight, not numbers

PARSEC 2.0

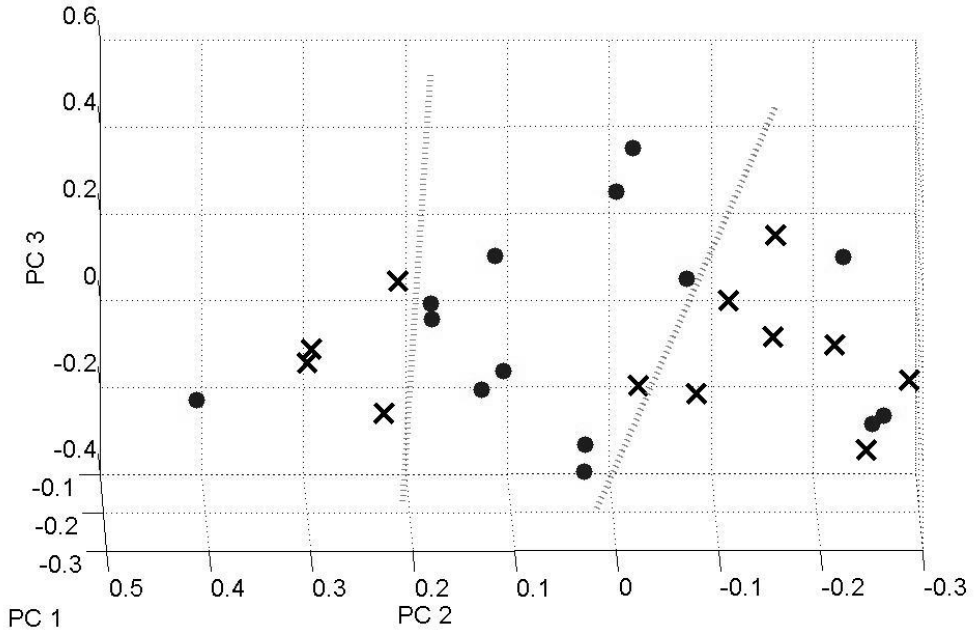
- One new workload: raytrace
- Much improved workloads: bodytrack, canneal, dedup and x264
- More parallelization models: Pthreads, OpenMP and Intel TBB
- Better portability, support for large-endian architectures
- Additional features for the framework

Workloads

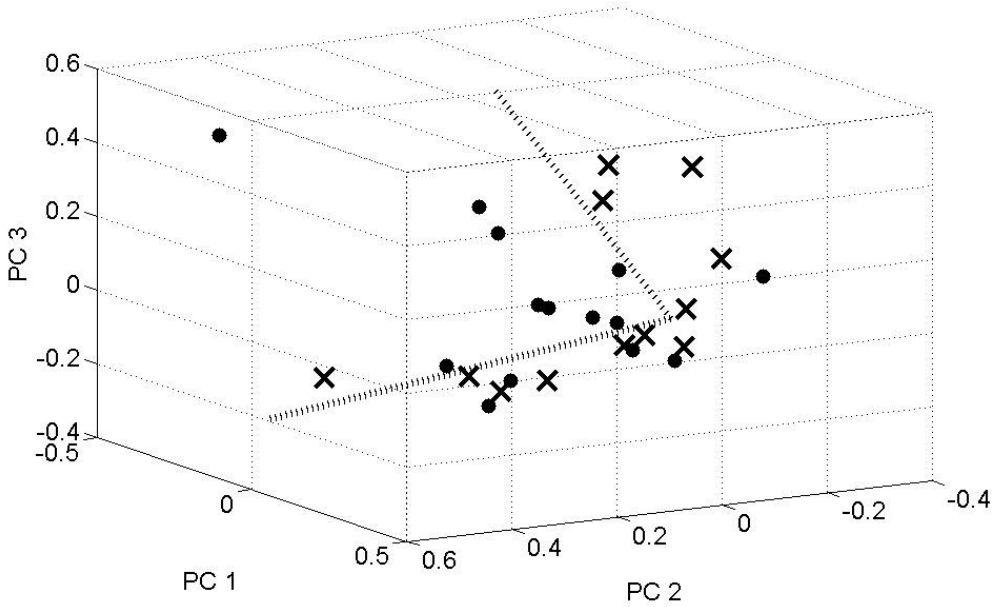
Program	Application Domain	Parallelization
Blackscholes	Financial Analysis	Data-parallel
Bodytrack	Computer Vision	Pipeline NEW!
Canneal	Engineering	Data-parallel NEW!
Dedup	Enterprise Storage	Pipeline
Facesim	Animation	Data-parallel
Ferret	Similarity Search	Pipeline
Fluidanimate	Animation	Data-parallel
Freqmine	Data Mining	Data-parallel
Raytrace NEW!	Visualization	Data-parallel
Streamcluster	Data Mining	Data-parallel
Swaptions	Financial Analysis	Data-parallel
Vips	Media Processing	Data-parallel
X264	Media Processing	Pipeline

PARSEC vs. SPLASH-2

Instruction Mix



Sharing



× PARSEC ● SPLASH-2

Statistical analysis shows significant differences. More details in [2].

You should expect different results

Blackscholes Overview

- Prices a portfolio of options with the Black-Scholes PDE
- Computational finance application (Intel)
- Synthetic input based on replication of 1,000 real options
- Coarse-granular parallelism, static load-balancing
- Small working sets, negligible communication

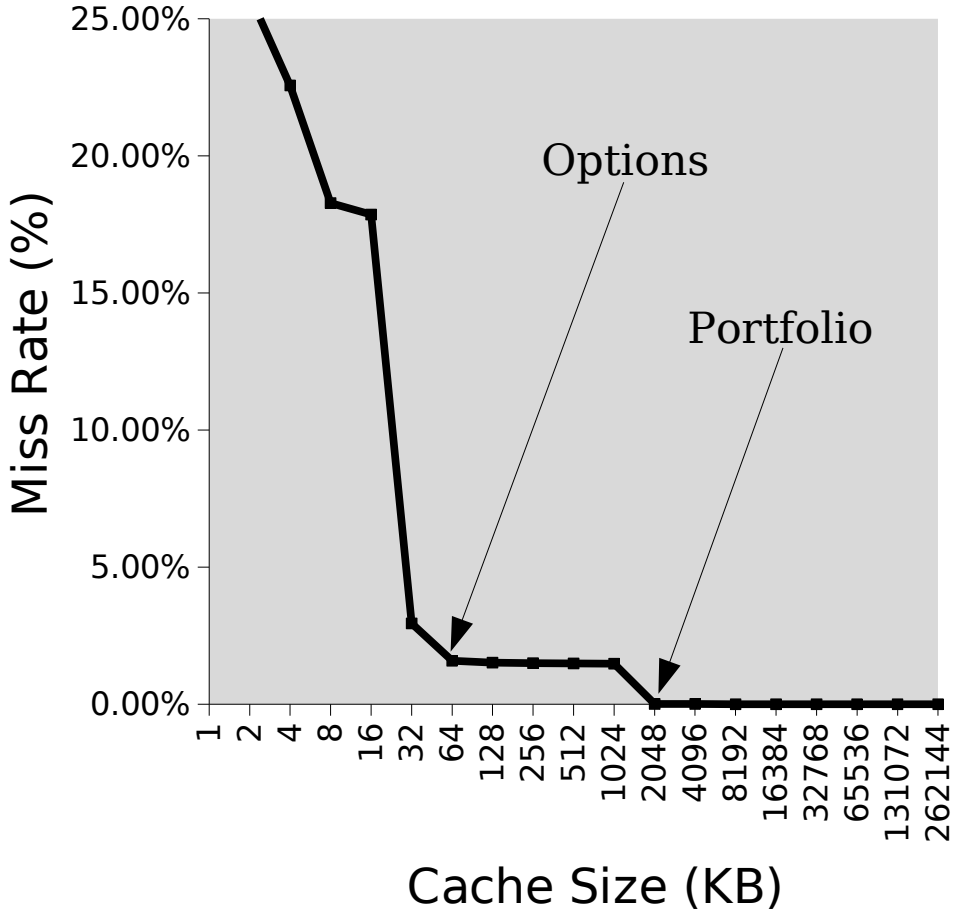
Blackscholes is the simplest of all PARSEC workload

Blackscholes Rationale

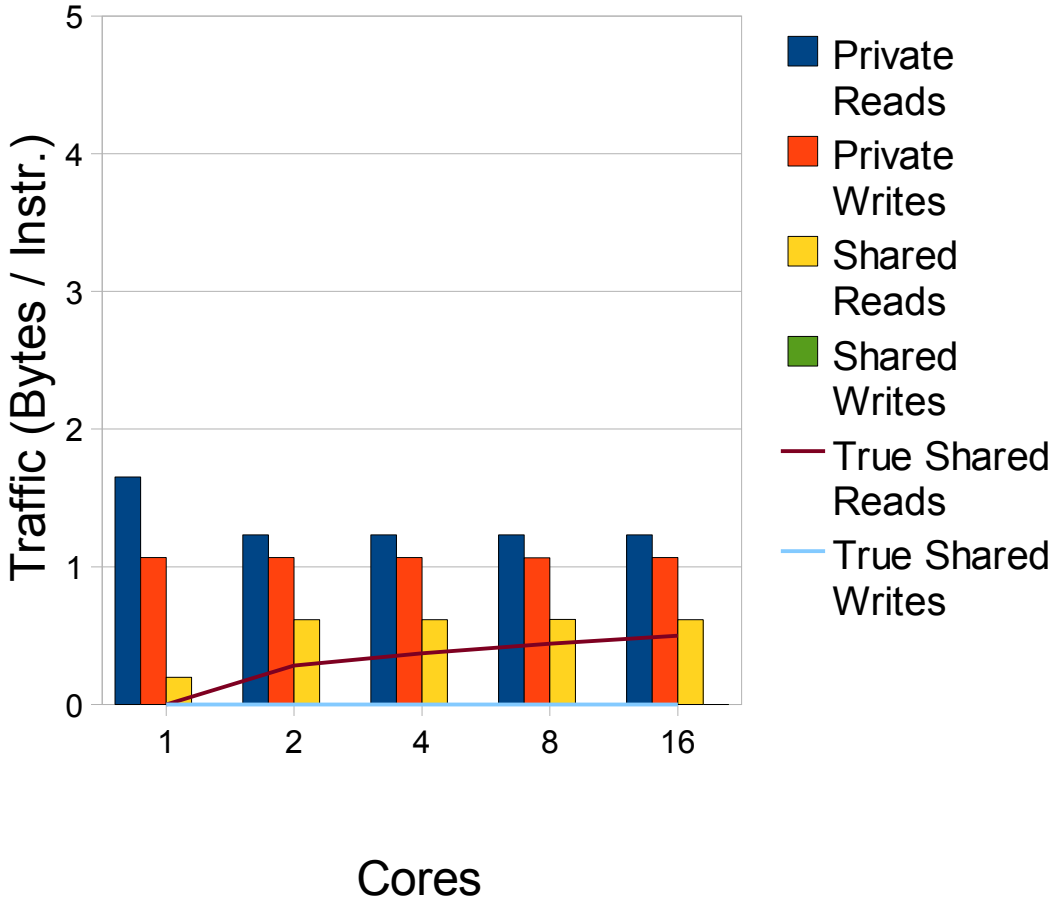
- Computers have become key technology for trading
- Derivatives are financial instrument with one of highest analytical requirements
- Blackscholes formula fundamental description of option behavior
- High demand for performance: Saving few milliseconds can earn lots of money

Blackscholes Characteristics

Working Sets



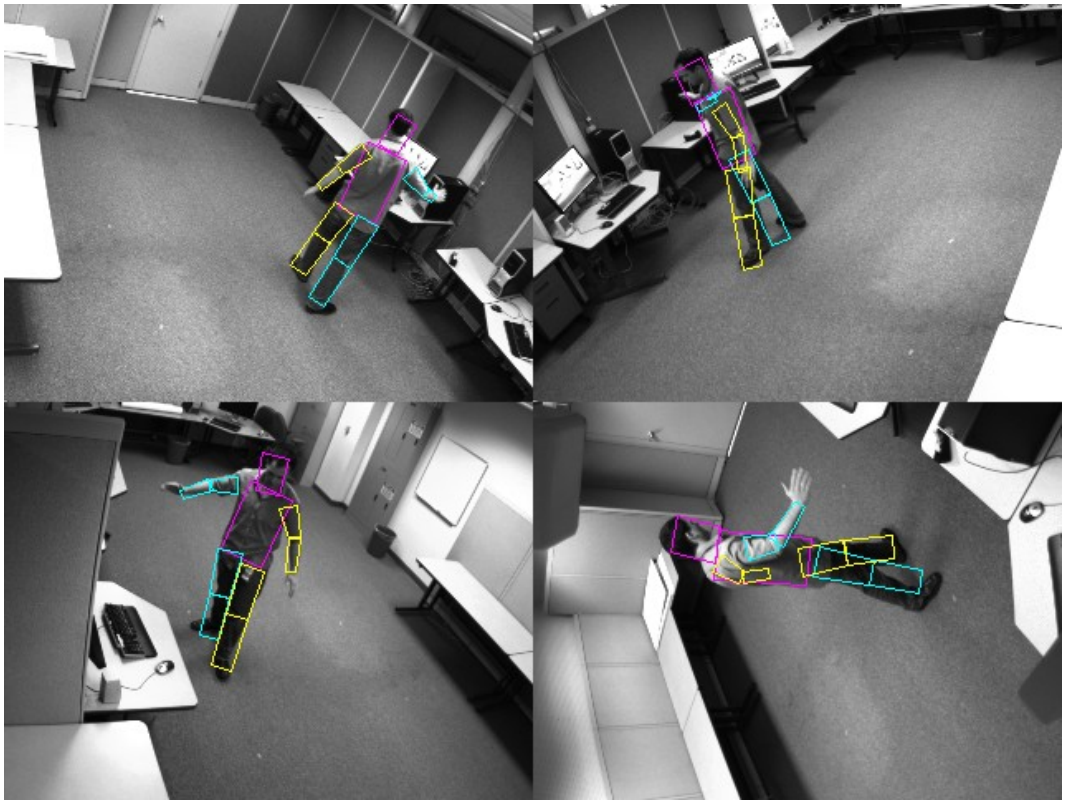
Cache Hits



Small working sets, negligible communication

Bodytrack Overview

- Tracks a markerless human body
- Computer vision application (Intel)
- Input is video feed from 4 cameras
- Medium-granular parallelism, dynamic load-balancing
- Pipeline and asynchronous I/O NEW!
- Medium working sets, some communication



Output of Bodytrack (Frame 1)

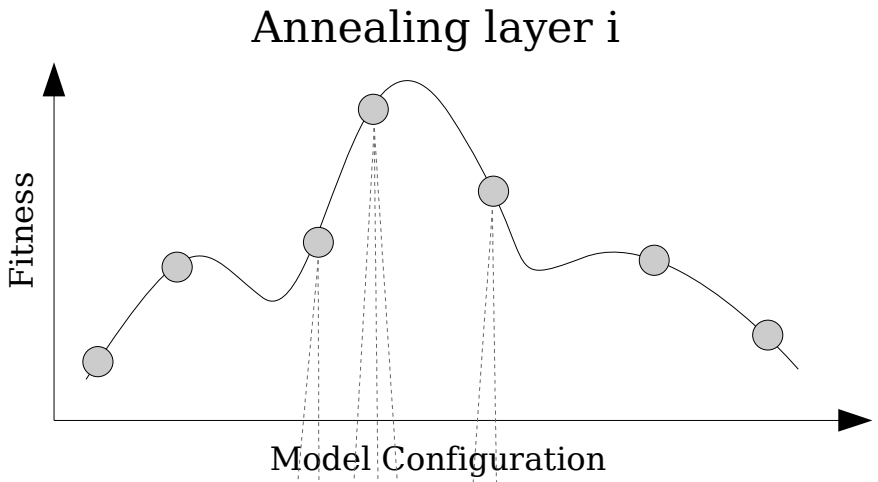
Bodytrack Rationale

- Machines increasingly rely on computer vision to interact with environment
- Often no aid available (e.g. Markers, constrained behavior)
- Must usually happen in real-time

Stanley, Winner of the DARPA Challenge 2005. Autonomous vehicle navigation requires real-time computer vision.

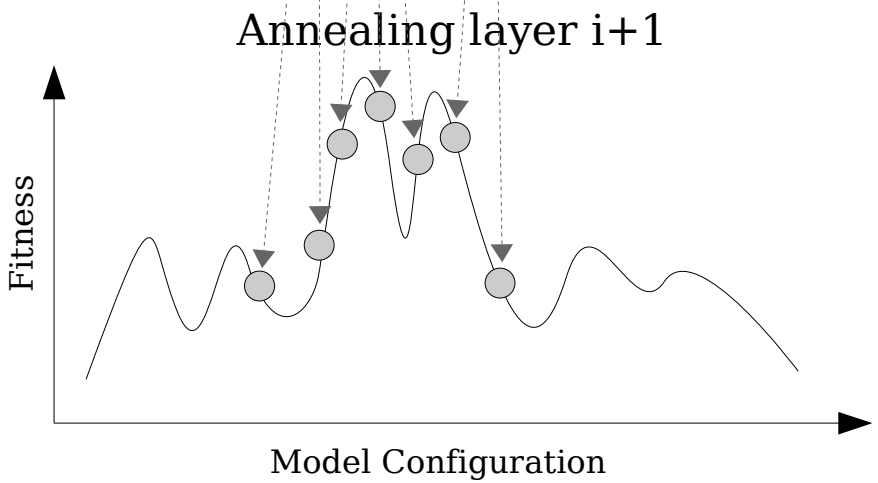
Bodytrack Idea

Bodytrack uses an annealed particle filter:



Compute fitness of all particles...

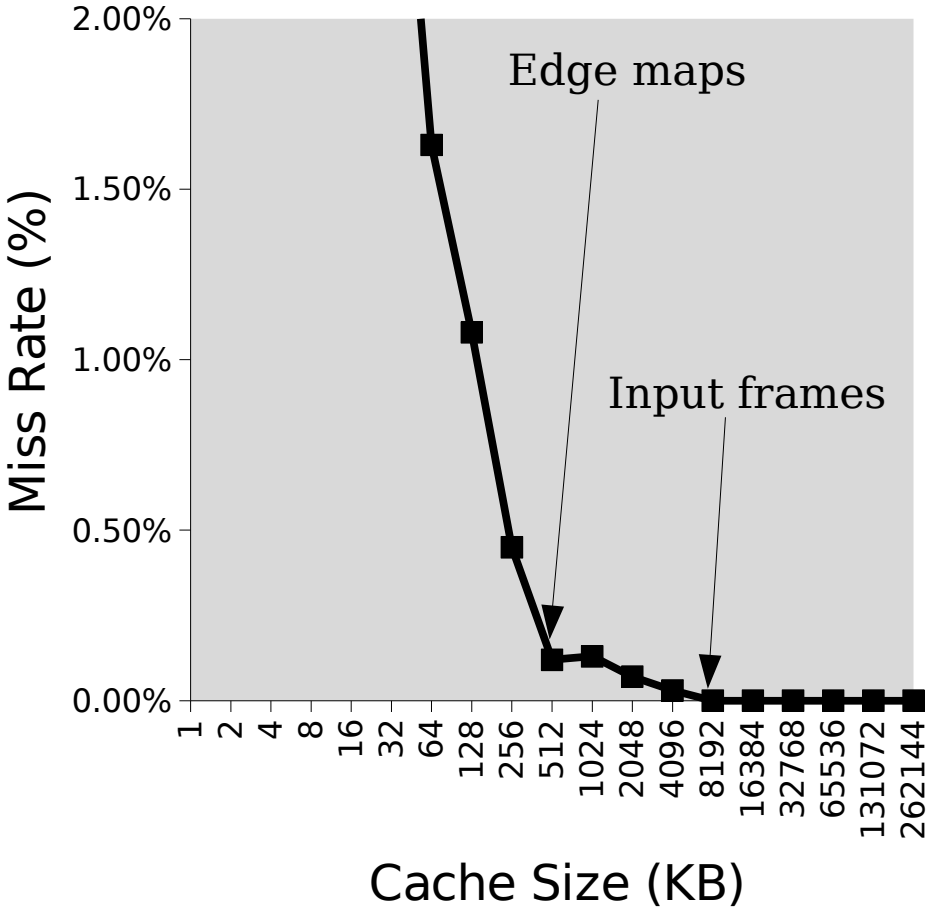
...then resample best particles...



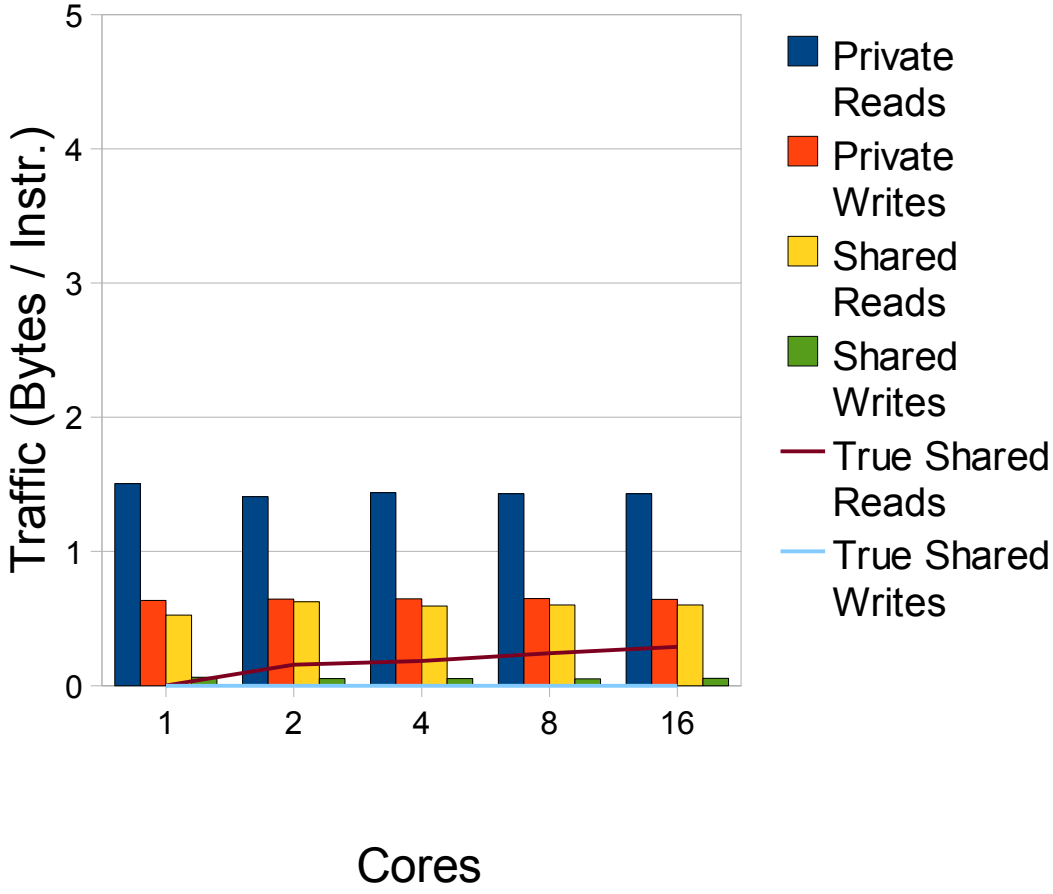
...to analyze interesting regions with more detail in another run.

Bodytrack Characteristics

Working Sets



Cache Hits



Medium working sets, some communication

Canneal Overview

- Minimizes the routing cost of a chip design with cache-aware simulated annealing
- Electronic Design Automation (EDA) kernel (Princeton)
- Input is a synthetic netlist
- Fine-granular parallelism, no problem decomposition
- Uses atomic instructions to synchronize
- Synchronization strategy based on data race recovery rather than avoidance
- Huge working sets, communication intensity only constrained by cache capacity.

Workload with most demanding memory behavior

Canneal Rationale

- Optimization is one of the most common types of problems.
- Place & Route is a difficult EDA challenge.
- Transistor counts continue to increase at an exponential rate.
- Simulated annealing allows to scale optimization cost by allowing incremental performance investments.

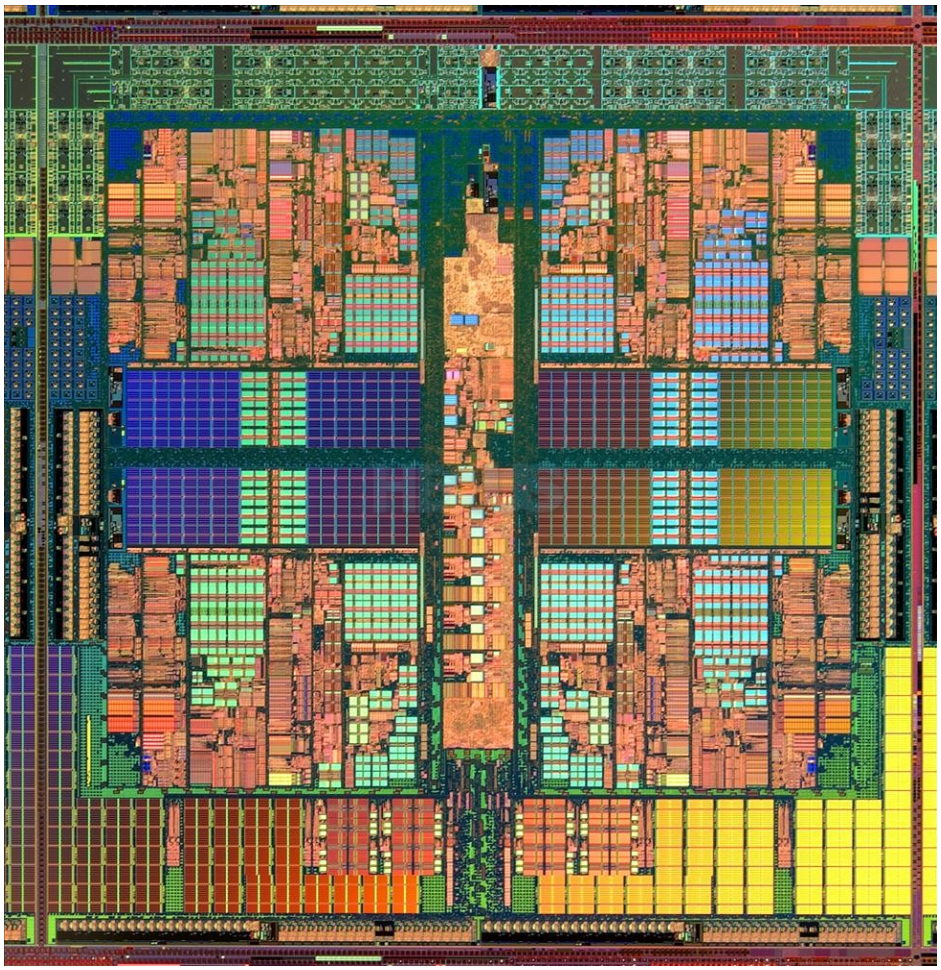
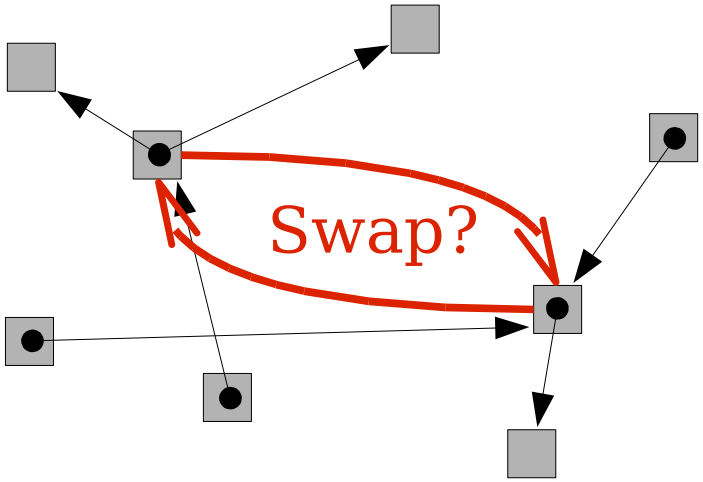


Photo of AMD's Barcelona quad-core CPU. It consists of about 463 million transistors.

Canneal Idea

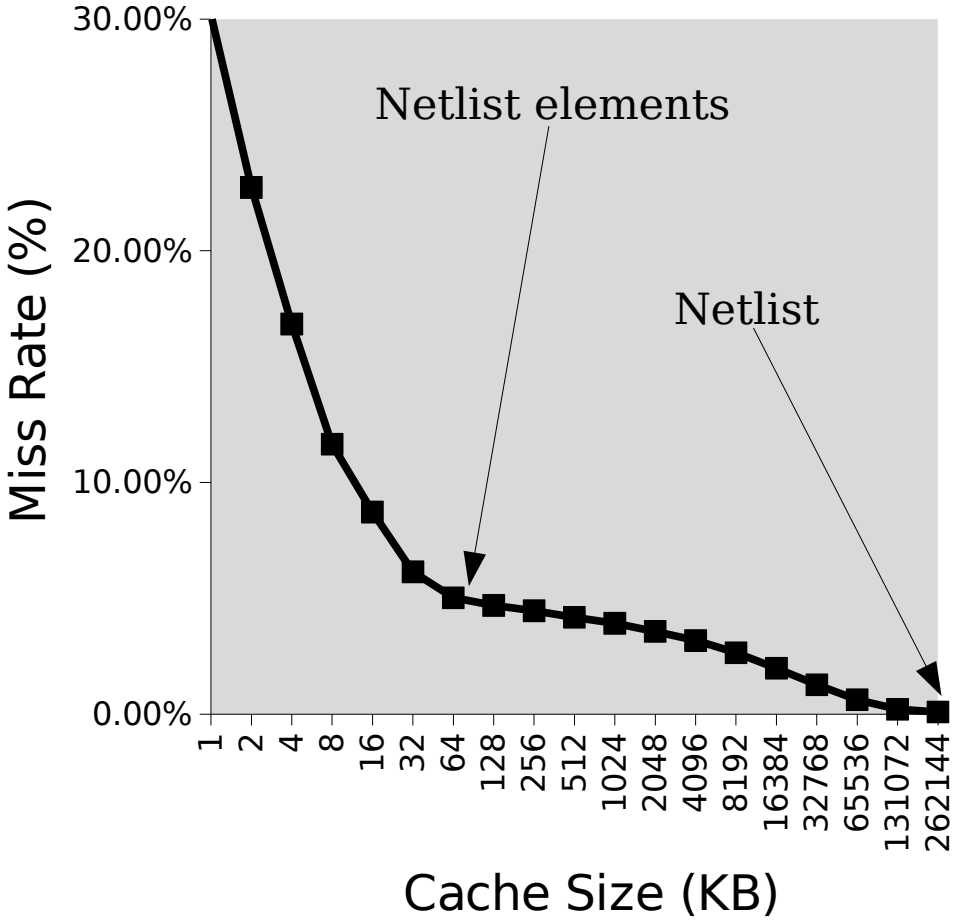


Be cache-aware: Discard only one element each round. Requires more swaps but less time.

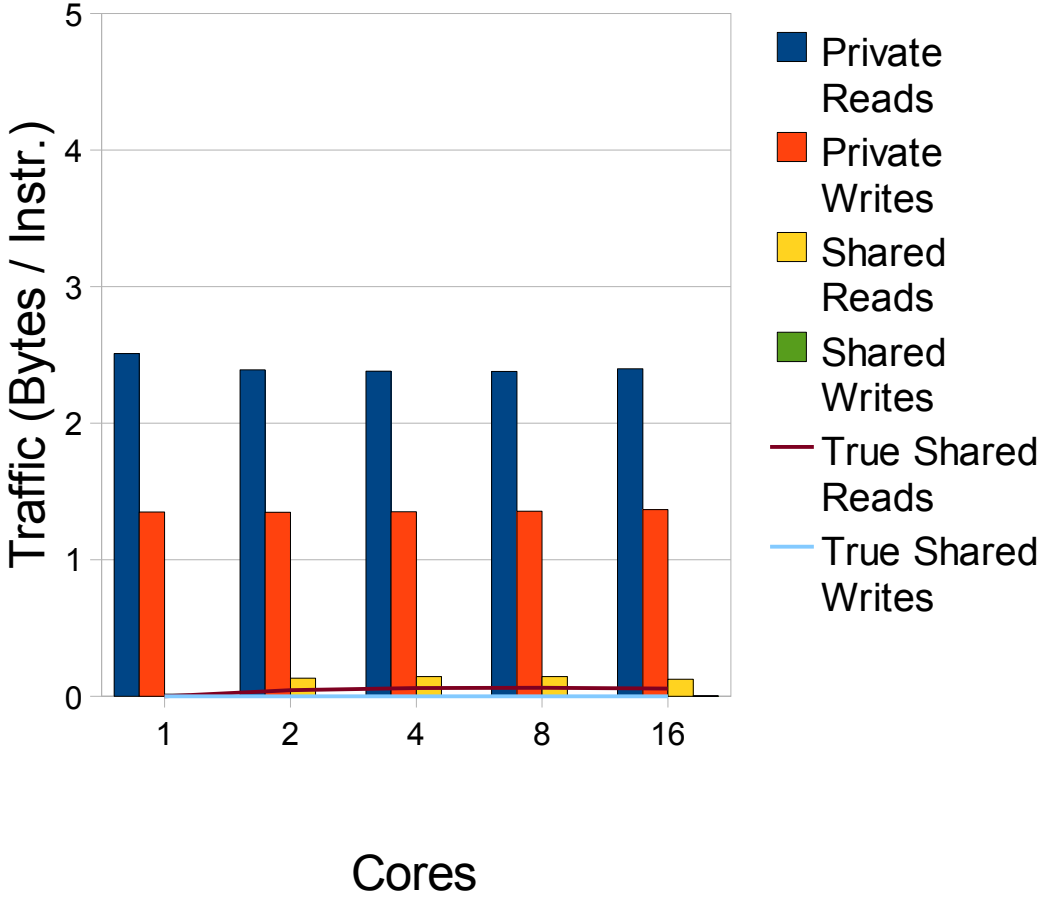
- Swap netlist elements to minimize routing cost
- Accept disadvantagous swaps with decreasing probability to escape local minimums and converge
- Swap atomically, but evaluate swaps concurrently
- Swap mistakes caused by races equal higher effective probability to accept bad swaps, recover from it automatically

Canneal Characteristics

Working Sets



Cache Hits



Huge working sets, communication limited by capacity

Dedup Overview

- Detects and eliminates redundancy in a data stream with a next-generation technique called 'deduplicaton'
- Enterprise storage kernel (Princeton)
- Input is an uncompressed archive containing various files
- Improved, more computationally intensive deduplication methods 

- More cache-efficient serial version 

- Pipeline parallelism with multiple thread pools
- Huge working sets, significant communication

Dedup Rationale

- Growth of world data keeps outpacing growth of processing power.
- This data has to be stored and transferred.
- Use cheap resources (processing power) to make more efficient use of scarce resources (storage & bandwidth).
- Already in use in commercial products.

datadomain

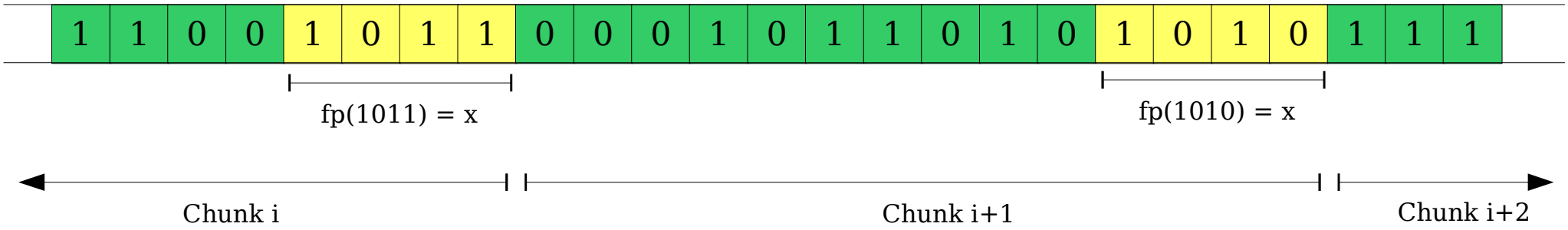
NetApp

EXAGRID
SYSTEMS

Next-generation storage and networking products already use data deduplication.

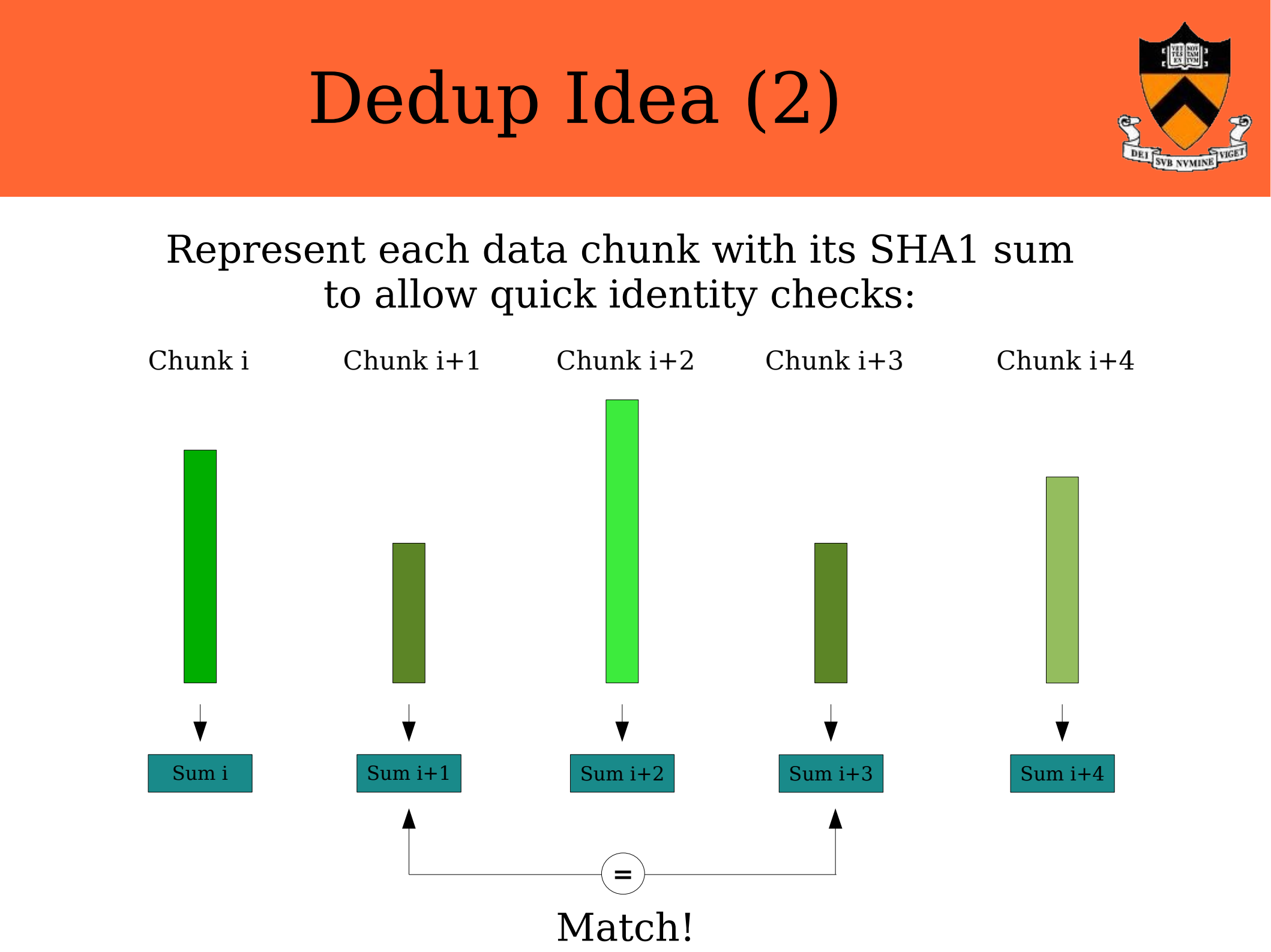
Dedup Idea (1)

Use rolling finger print to break data stream into chunks:



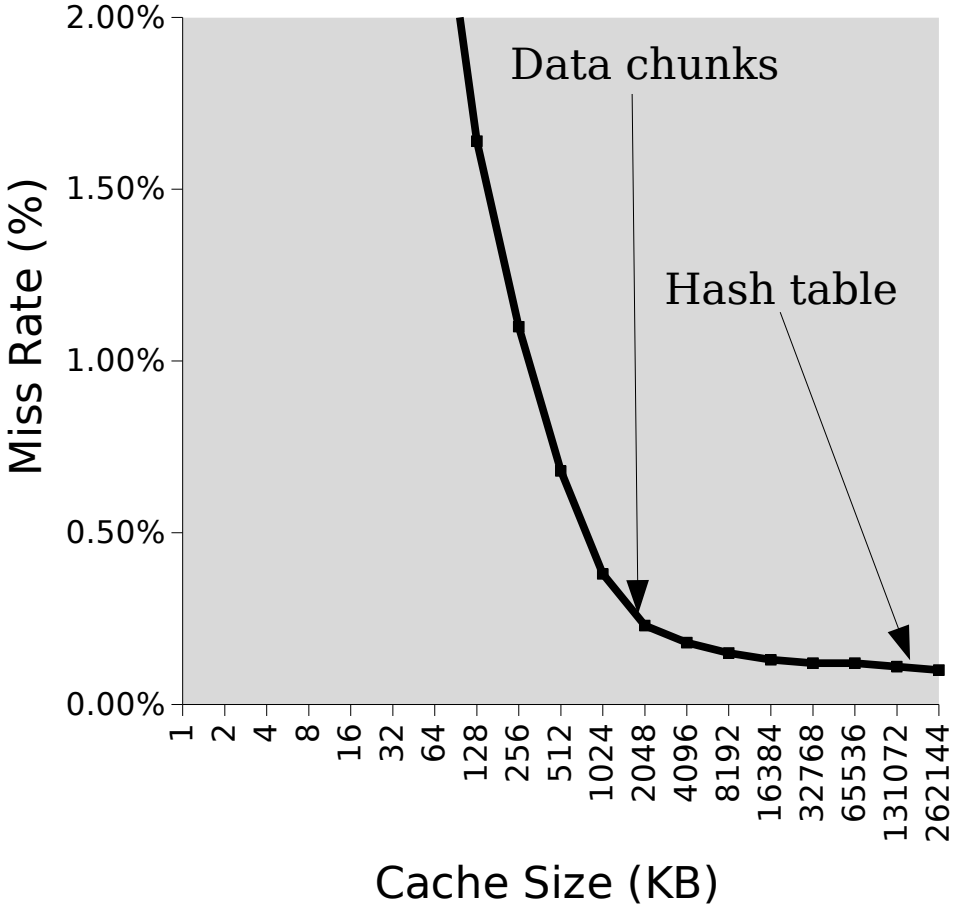
- Results in independent data chunks of variable size
- Few bits of finger print used, does not guarantee identity
- If a bit pattern causes a split then the data stream will be split at all of its occurrences
- Guarantees that fragmenting a data stream will not obscur identical bit patterns

Dedup Idea (2)

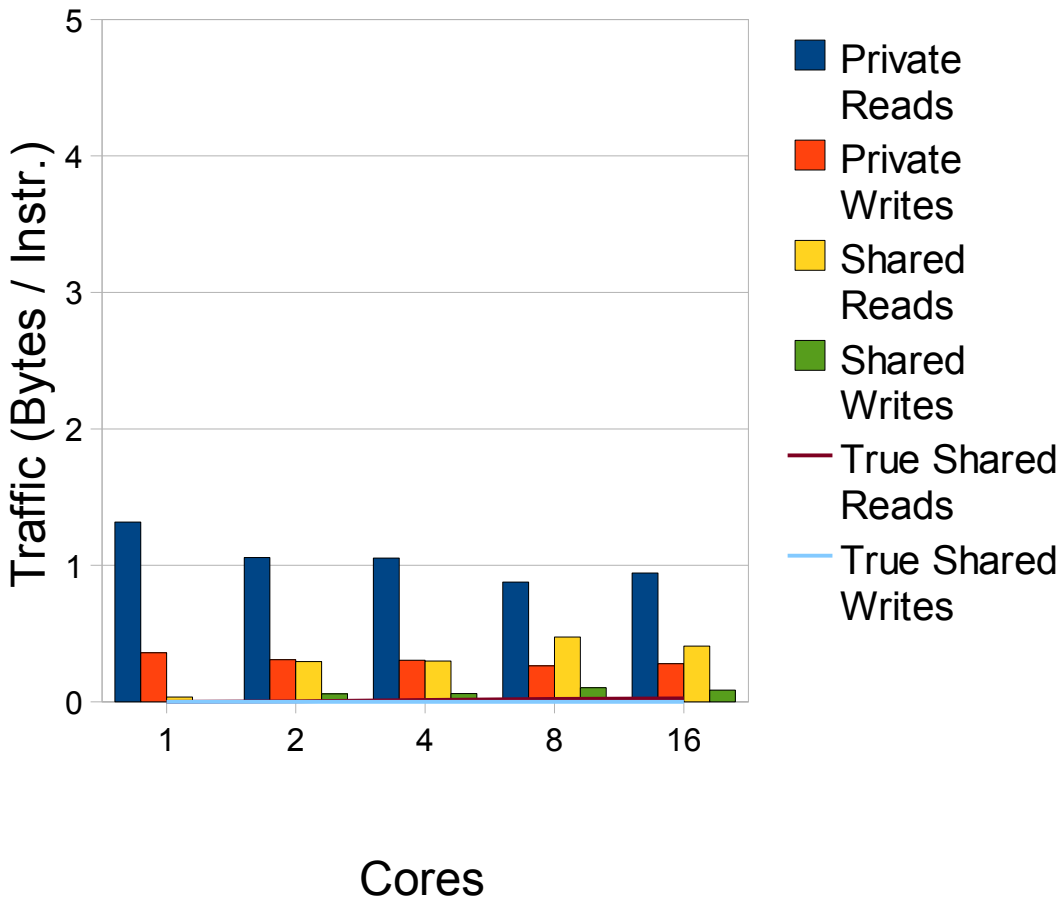

Represent each data chunk with its SHA1 sum
to allow quick identity checks:
Chunk i
Chunk i+1
Chunk i+2
Chunk i+3
Chunk i+4
Sum i
Sum i+1
Sum i+2
Sum i+3
Sum i+4
=
Match!

Dedup Characteristics

Working Sets



Cache Hits



Huge working sets, some communication

Facesim Overview

- Simulates motions of a human face for visualization purposes

- Computer animation application (Intel + Stanford)
- Input is a face model and a series of muscle activations
- Coarse-grained parallelism, similarities to HPC programs
- Large working sets, some sharing

Source: Eftychios Sifakis et al.

Facesim creates visually realistic animations of a human face

Facesim Rationale

- Video games and other interactive animations require visualization of realistic faces in realtime
- Challenging problem, humans evolved to perceive finest details in a face
- Physical simulation gives excellent results, but is computationally very challenging
- Technology already in use for movie productions (e.g. Pirates of the Caribbean 3)

Faces are an integral part of contemporary games. Screenshot of Codemasters' "Overlord: Raising Hell" (2008).

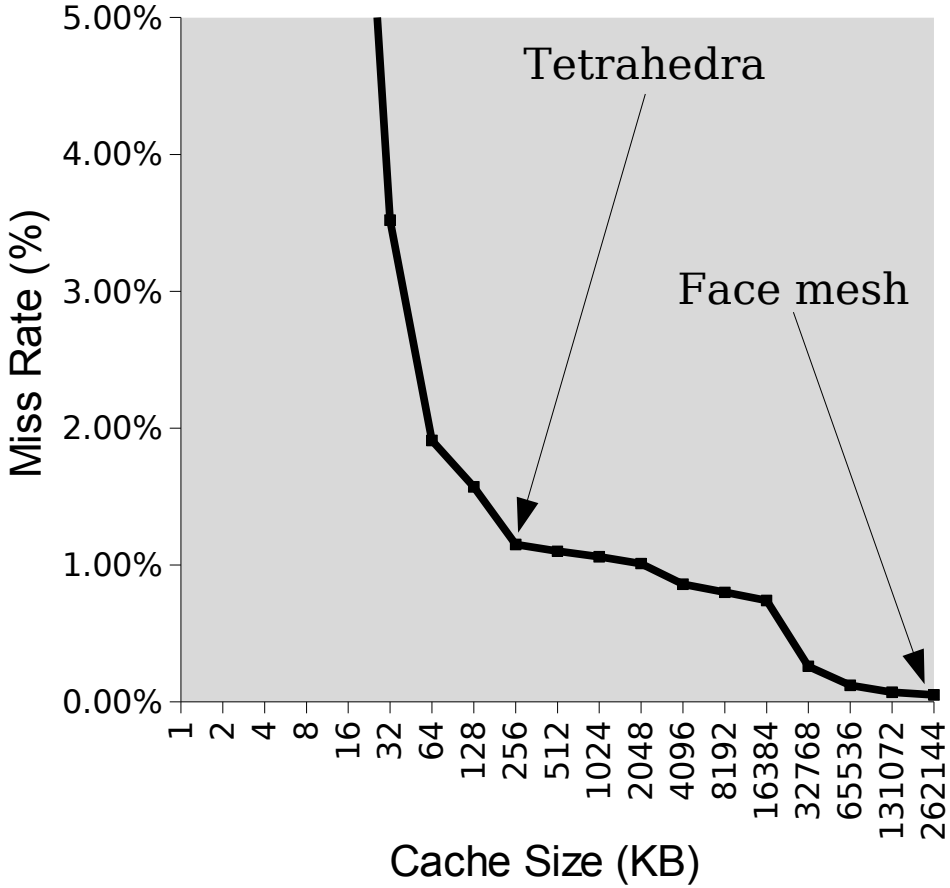
Facesim Demonstration

Source: Eftychios Sifakis et al.

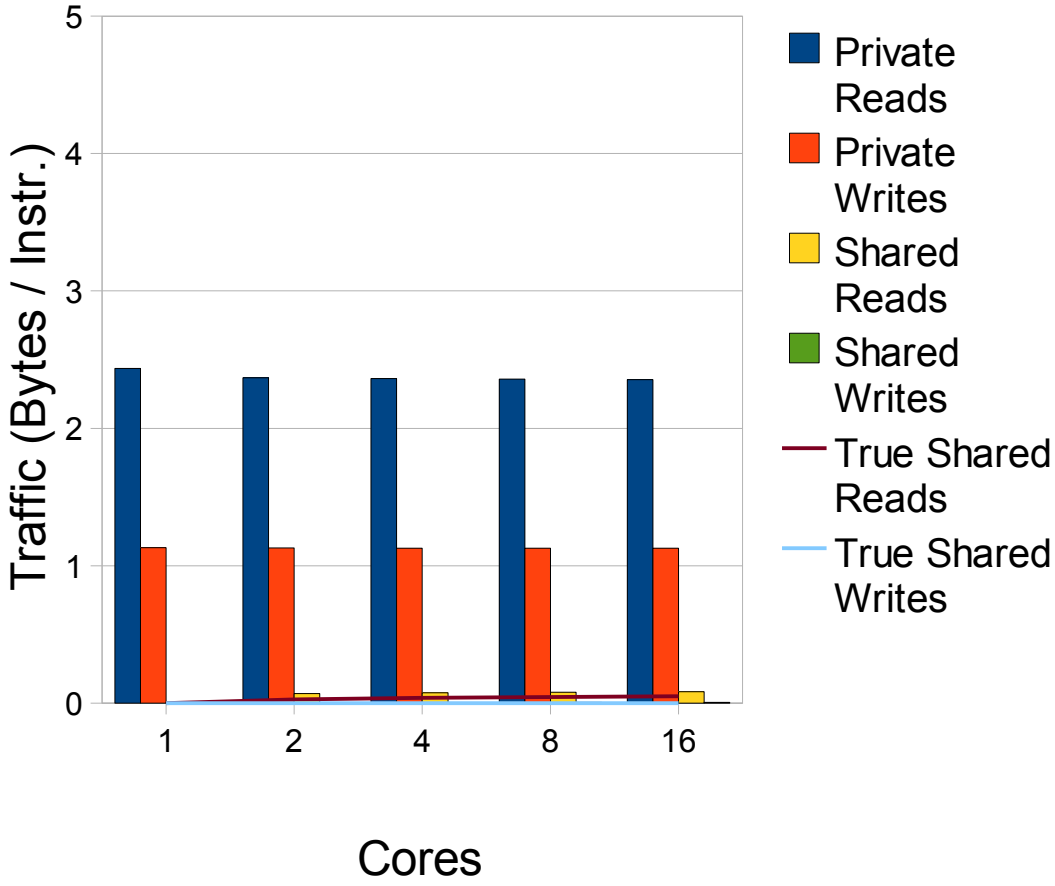
Video: Face deformation by lollipop

Facesim Characteristics

Working Sets



Cache Hits



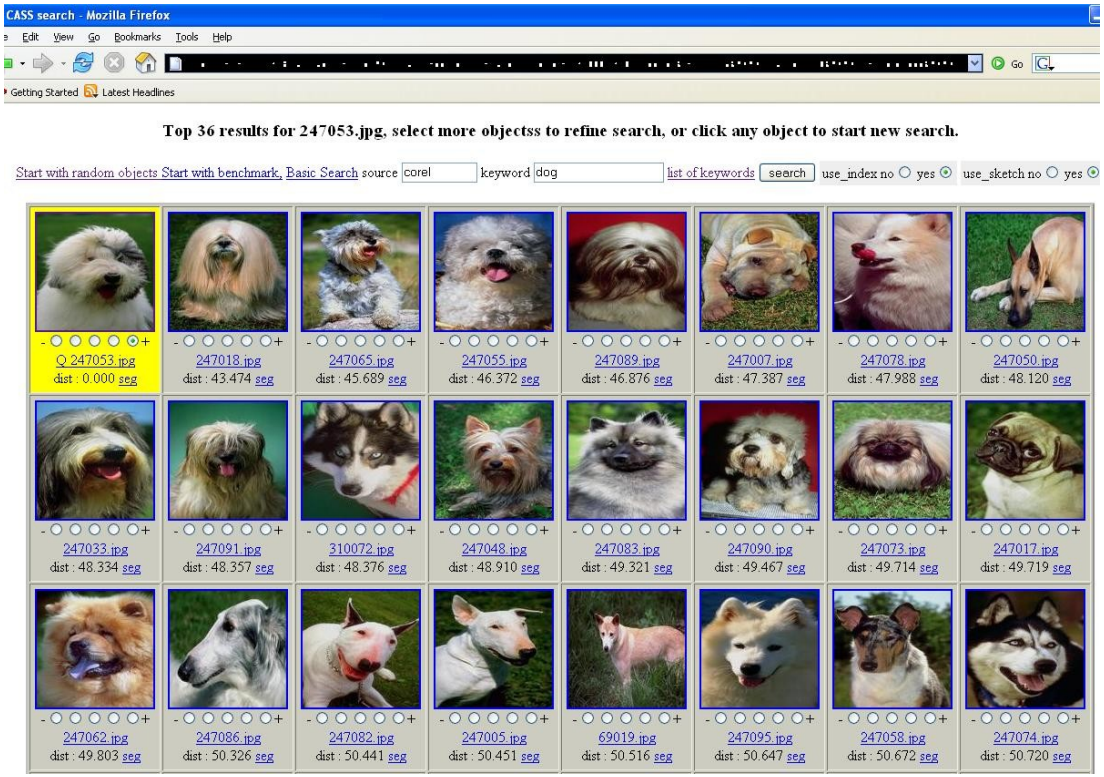
Large working sets, some sharing

Ferret Overview

- Search engine which finds a set of images similar to a query image by analyzing their contents
- Server application for content-based similarity search of feature-rich data (Princeton)
- Input is an image database and a series of query images
- Pipeline parallelism with multiple thread pools
- Huge working sets, very communication intensive

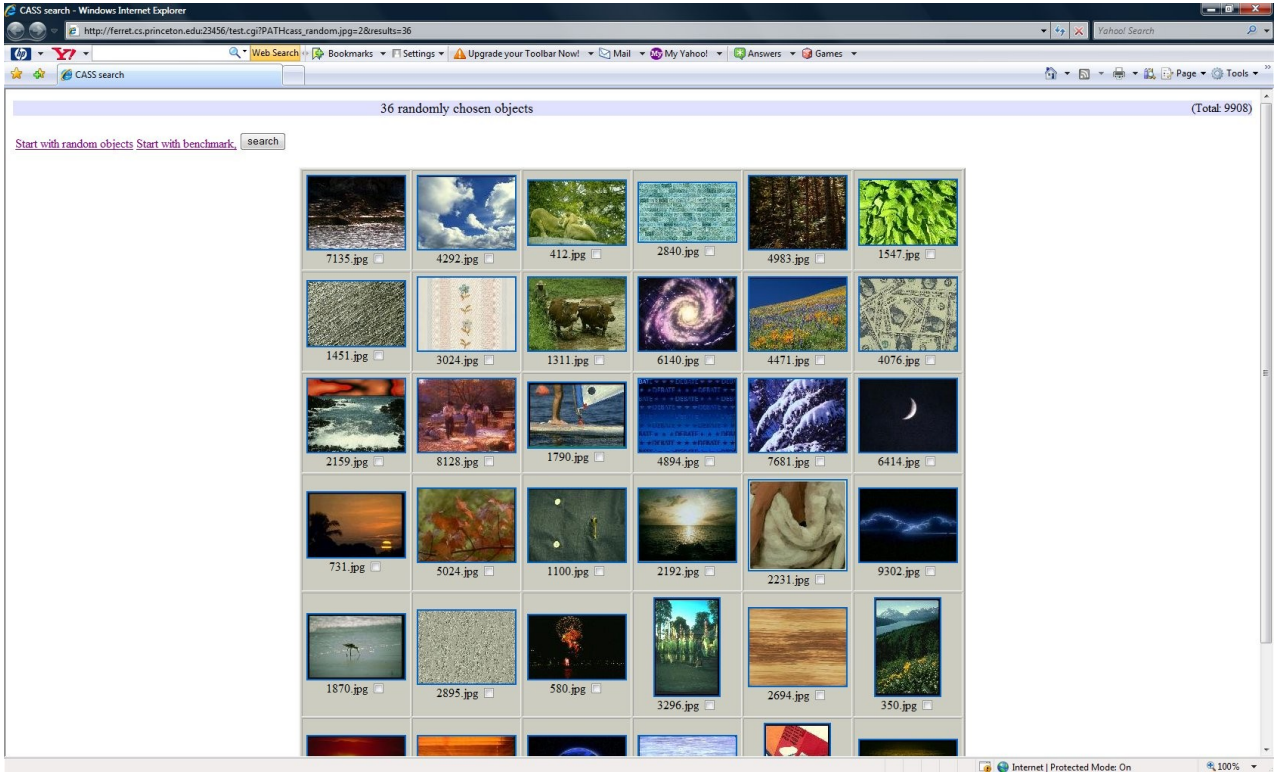
Ferret Rationale

- Growth of world data requires methods to search and index it
- Noise and minor variations frequently make same content appear slightly different
- Traditional approaches using key words are inflexible and don't scale well
- Computationally expensive



A web interface for image similarity search.

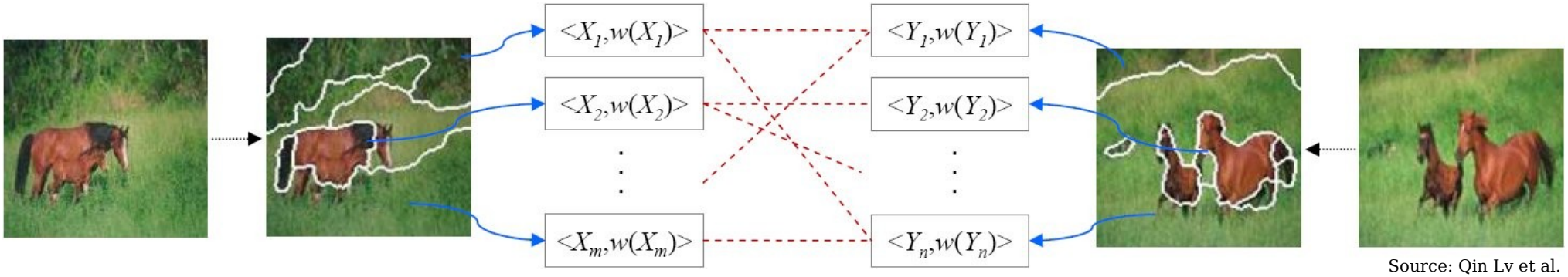
Ferret Demonstration



http://www.cs.princeton.edu/cass/demos.htm

Ferret Idea

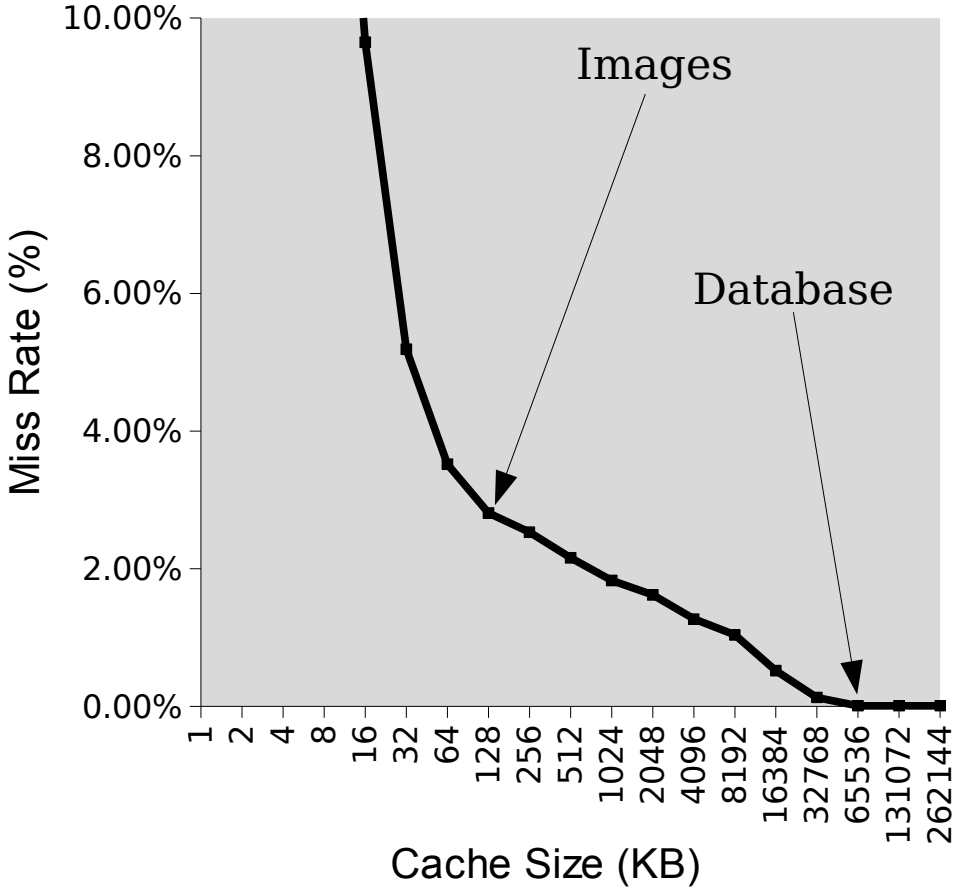
Use pair-wise comparison of image segments to compute score:



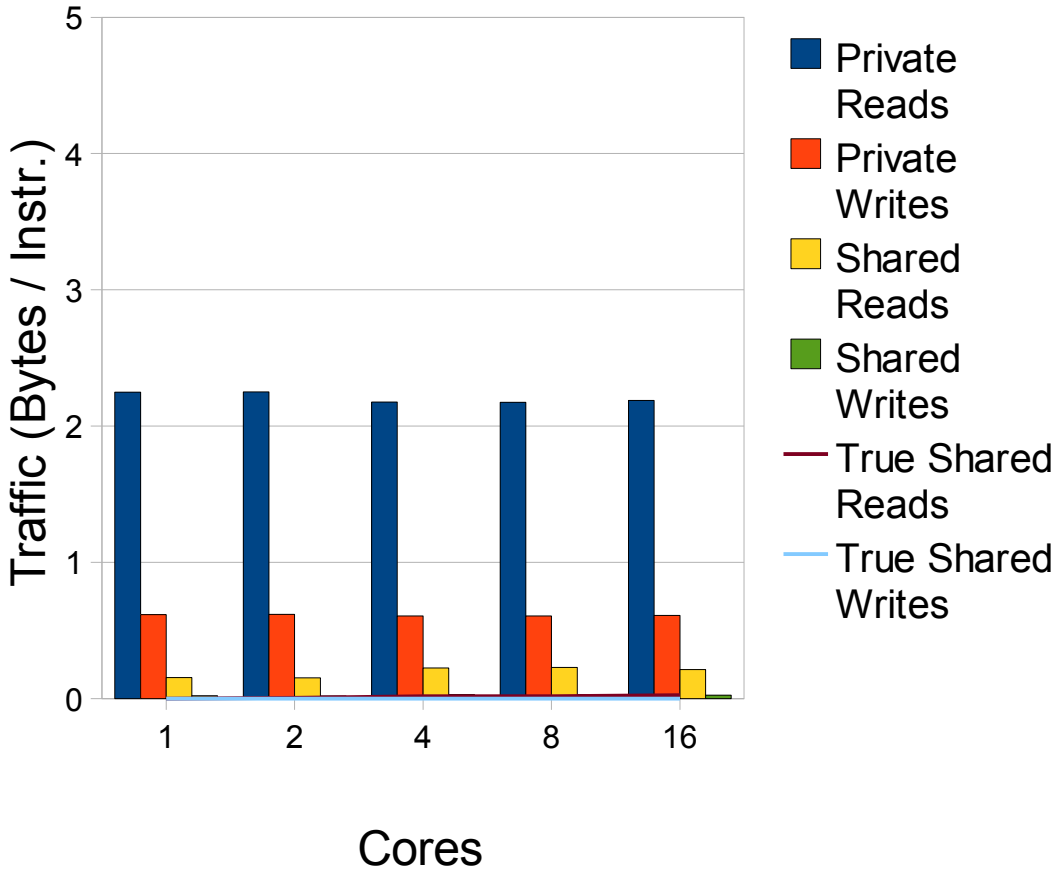
- Break all images up into segments in order to be able to distinguish different objects in image
- Represent segments with feature vectors which mathematically describe content of segment

Ferret Characteristics

Working Sets



Cache Hits



Huge working sets, very communication intensive

Fluidanimate Overview

- Simulates the underlying physics of fluid motion for realtime animation purposes with SPH algorithm
- Computer animation application (Intel)
- Input is a list of particles
- Coarse-granular parallelism, static load balancing
- Large working sets, some communication

Fluidanimate Rationale

- Physics simulations allows significantly more realistic animations
- Highly demanded feature for games
- Fluid animation one of most challenging effects
- Already beginning to get used in games

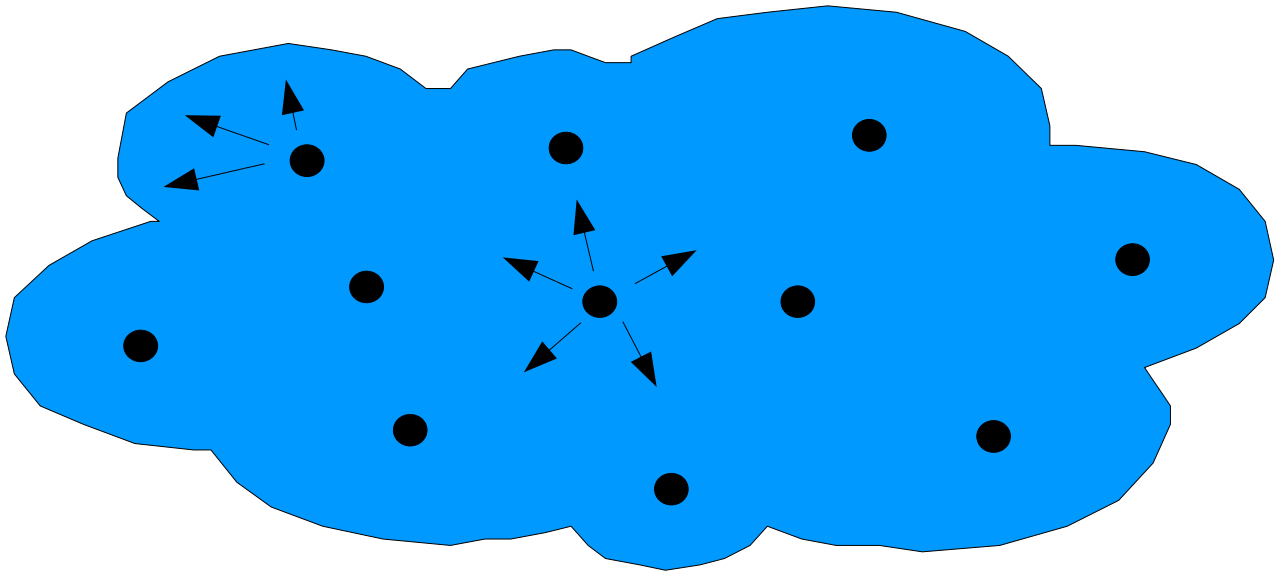
Advanced physics effects are already starting to get used in games: Tom Clancy's Ghost Recon Advanced Warfighter (2006) with (left) and without (right) PhysX effects.

Fluidanimate Demonstration

Video: Cell Factor (2007) with PhysX

Fluidanimate Idea

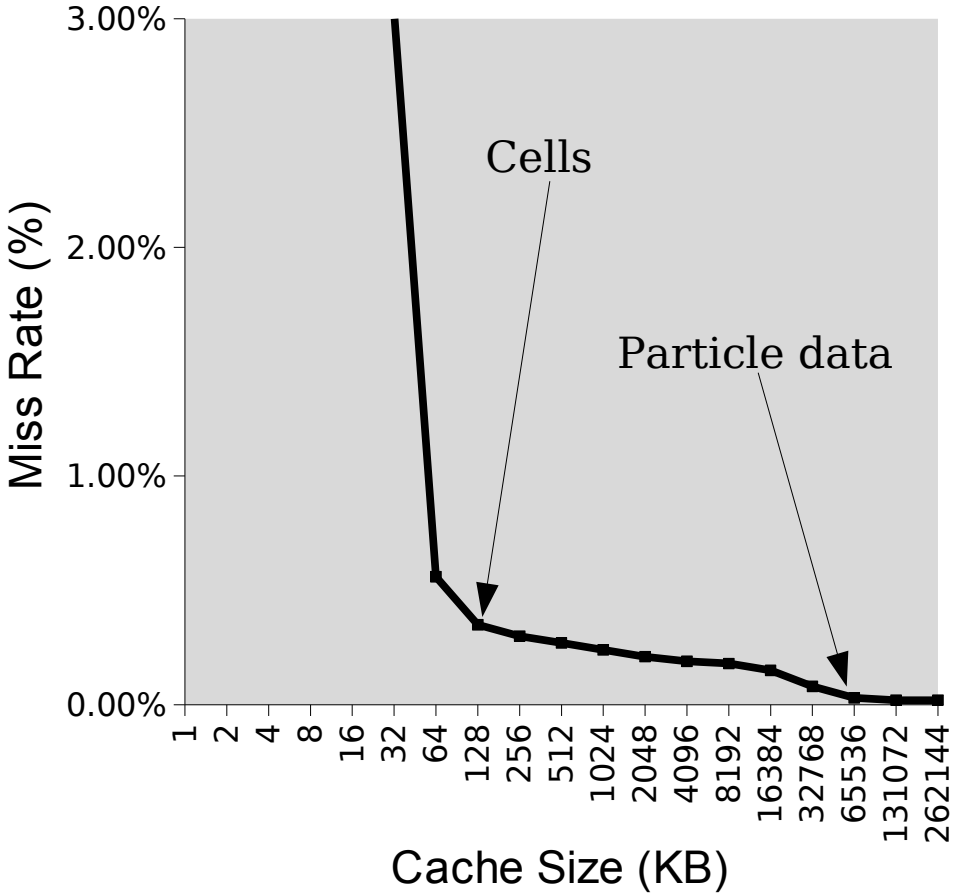
Approximate continuous fluid with discrete particles:



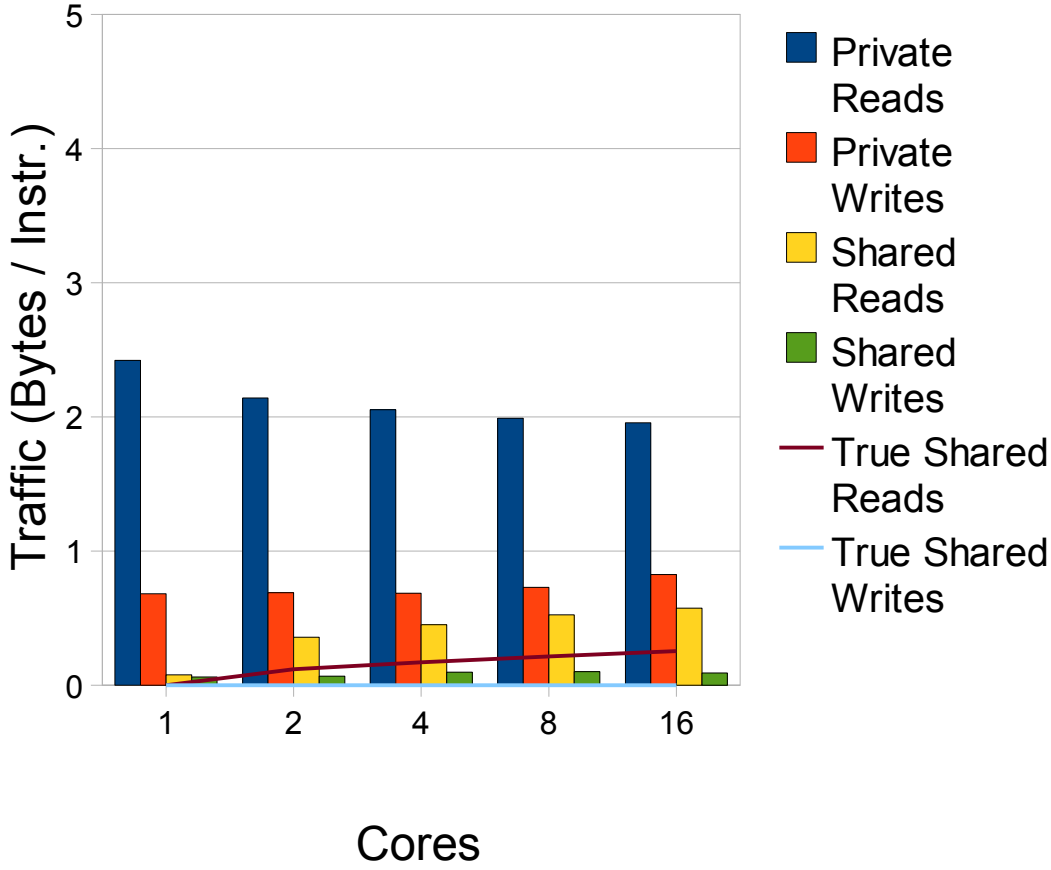
- Dense particles indicate high fluid pressure
- Particles interact with neighboring particles
- Need another algorithm to reconstruct fluid surface

Fluidanimate Characteristics

Working Sets



Cache Hits



Large working sets, some communication

Freqmine Overview

- Identifies frequently occurring patterns in a transaction database
- Data mining application (Intel + Concordia)
- Input is a list of transactions
- Medium-granular parallelism, parallelized with OpenMP
- Huge working sets, some sharing

Freqmine Rationale

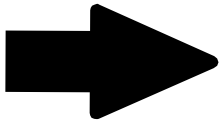
- Increasing amounts of data need to be analyzed for patterns
- Applies to many different areas such as marketing, computer security or computational biology
- Requirements for computational processing power virtually unlimited in practice

Frequent Itemset Mining is already used e.g. for e-commerce (Screenshot: Amazon.com).

Freqmine Idea

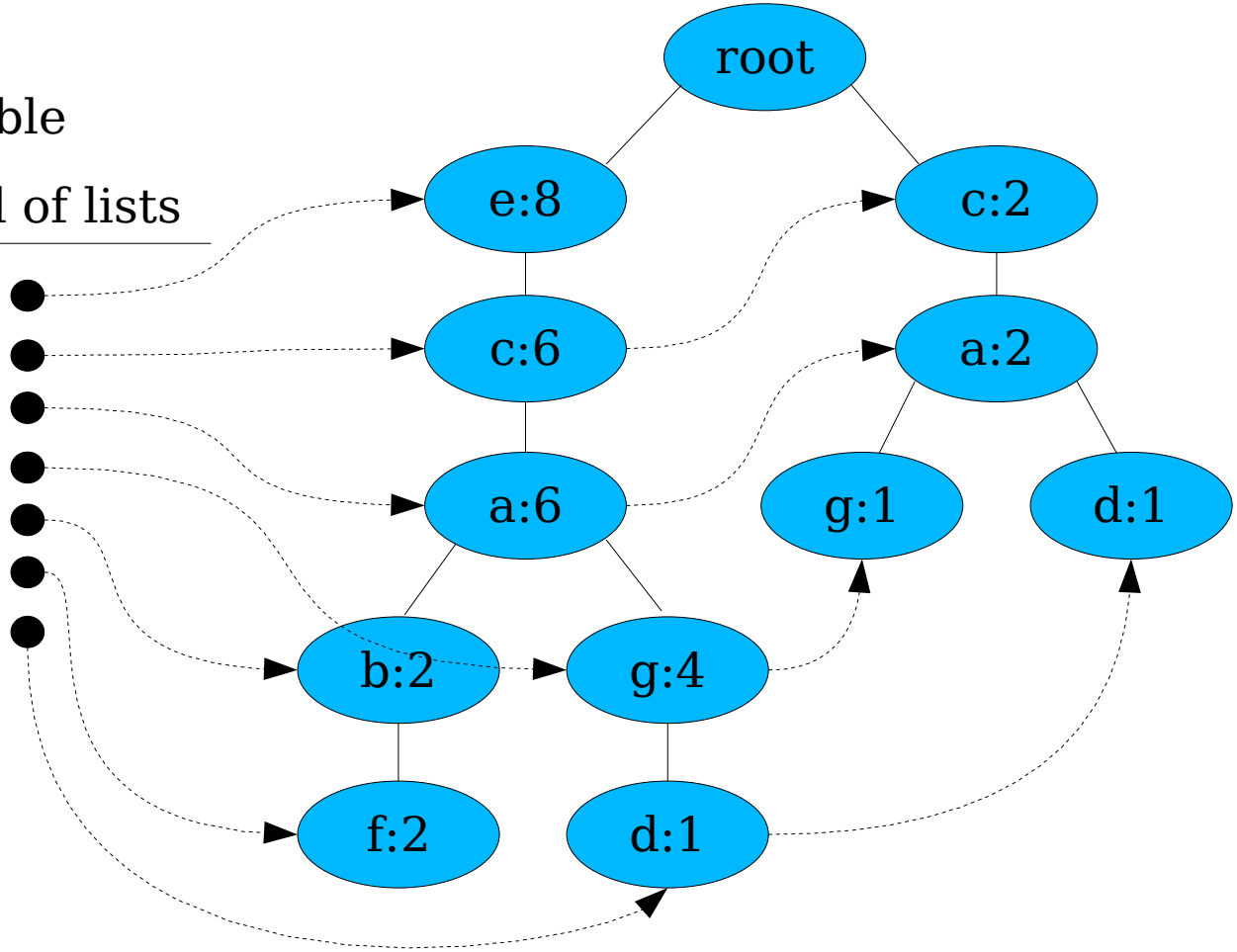
Store transaction database as a FP-tree:

abcefo
acg
ei
acdeg
acegl
ej
abcefp
acd
acegm
acegn



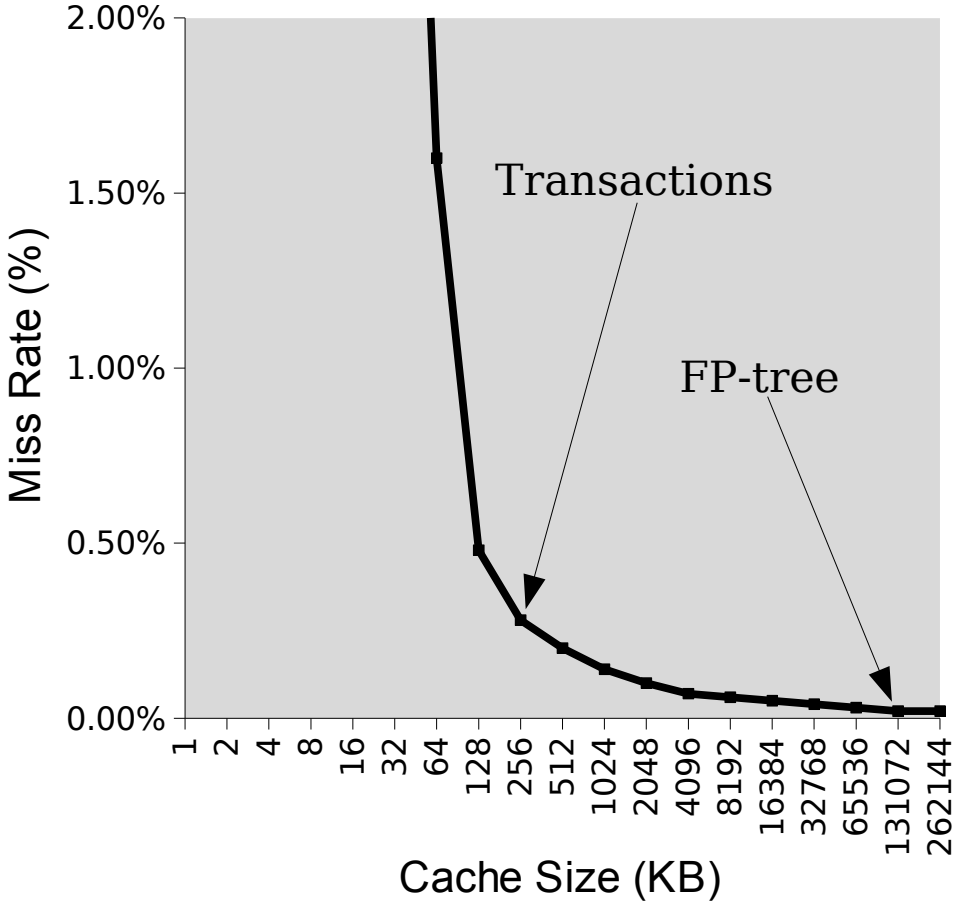
Header table

Item	Head of lists
e:8	
c:8	
a:8	
g:5	
b:2	
f:2	
d:2	

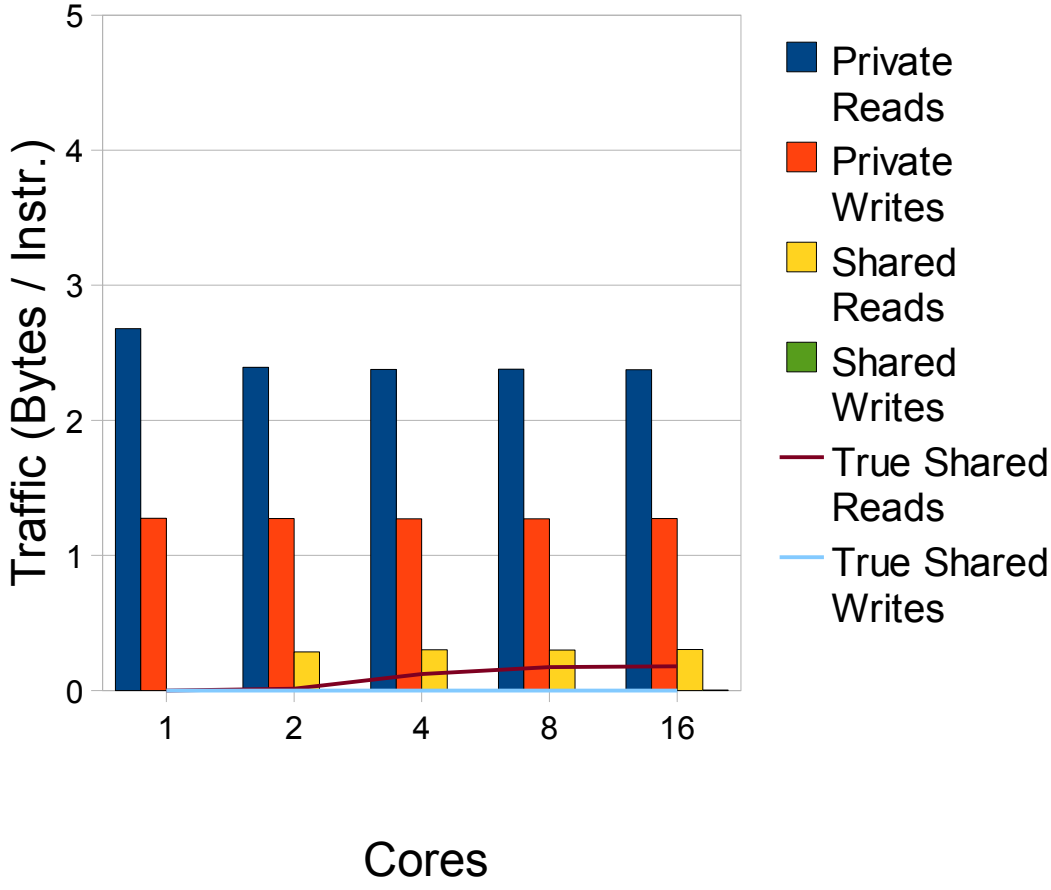


Freqmine Characteristics

Working Sets



Cache Hits



Huge working sets, some sharing

Raytrace Overview

- Uses physical simulation for visualization
- Computer animation application (Intel)
- Input is a complex object composed of many triangles
- Fine-granular parallelism, dynamic load balancing
- Large working sets, little communication, significant data sharing

Source: Stanford University

Native input for raytrace. (10 million polygons)

Raytrace Rationale

- Physics simulations allows accurate visualizations with realistic 3D graphics
- Realistic effects possible without tricks (shadows, reflections, refractions, etc.)
- Simpler development of games at the cost of more expensive computations

May 23, 2008 10:12 AM PDT

Nvidia buys ray-tracing tech company RayScale

Posted by Brooke Crothers Font size Print E-mail Share Post a comment

Nvidia confirmed Friday that it has acquired RayScale, a small company that develops ray-tracing technology. Financial terms of the deal have not been disclosed.

rayscale

Ray tracing has been mentioned frequently by Intel over the last six months. An Intel blog titled "**Real Time Ray-Tracing: The End of Rasterization?**" and later comments by Intel executives that the company **is looking at doing ray tracing on its processors** set the stage for debate on the viability of ray tracing in mainstream gaming.

PC graphics technology today uses **rasterization**. (**A discussion of ray tracing vs. rasterization**.)

Ray Tracing is a technique for rendering three-dimensional graphics with extremely complex light interactions, allowing the creation of transparent surfaces and shadows, for example, with stunning photorealistic results.

Ray tracing is a highly parallel process. And the GPU (graphics processing unit) provides high level of parallelism, according to Nvidia officials speaking at a conference on Thursday. The GPU has special function units that were desgined for doing graphics operations that are perfect for ray tracing, said Nvidia Chief Scientist David Kirk.

At the conference, Kirk and RayScale scientists discussed "GPU ray tracing." It's not clear how soon this technology would be used commercially by Nvidia.

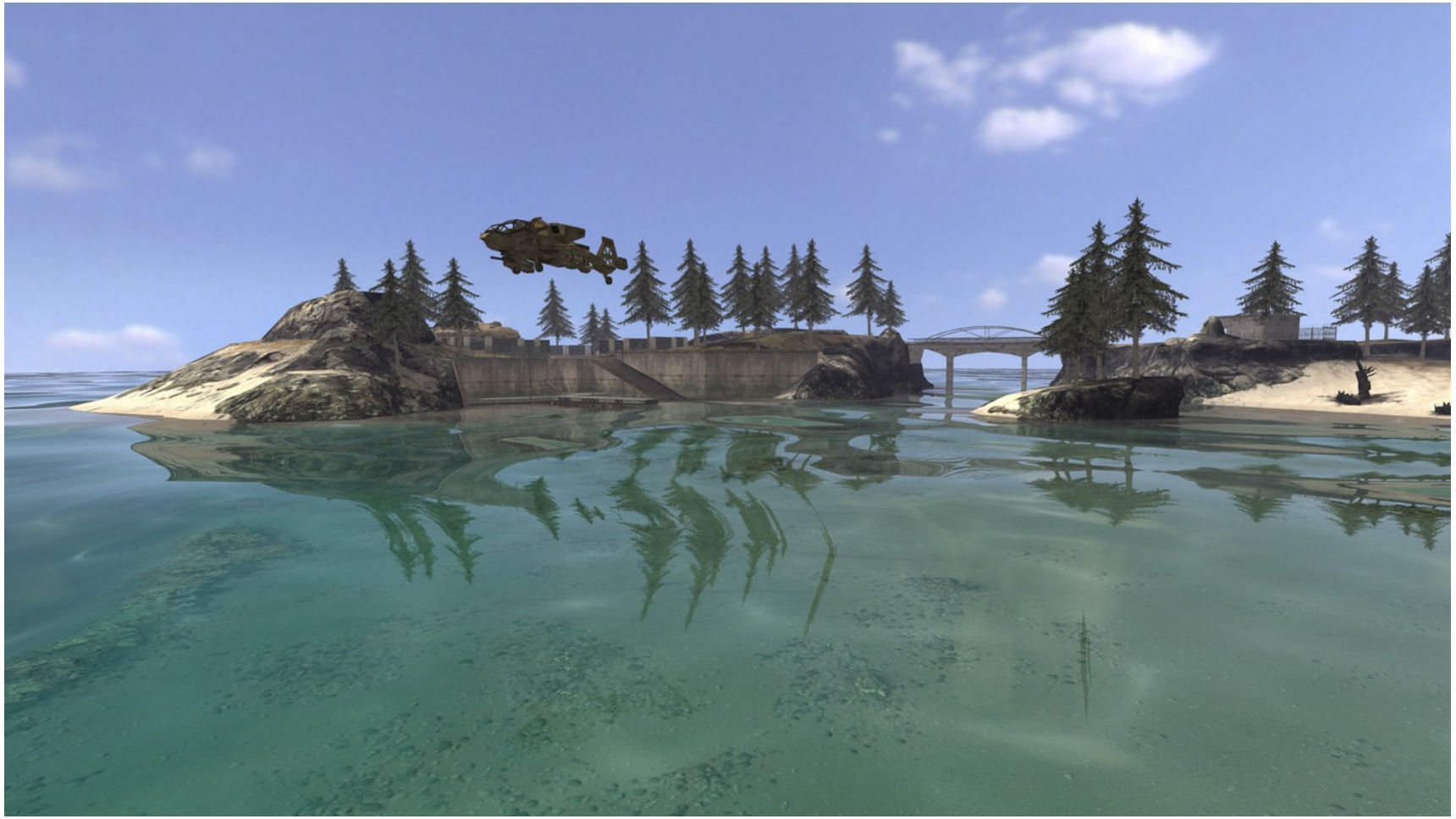
RayScale, which provides interactive ray tracing and photo-realistic rendering solutions, says its technologies "dramatically increase the speed and realism at which graphics professionals can produce high quality three-dimensional computer graphics and photorealistic computer images."

RayScale is a product of the decade-long interactive ray-tracing research at the University of Utah, **according to RayScale**.

At the Intel Developer Forum in Shanghai in April, Senior Intel Vice President Patrick Gelsinger spelled out Intel's

Major companies have started to invest into ray tracing (Source: cnet, May 2008)

Raytrace Demonstration

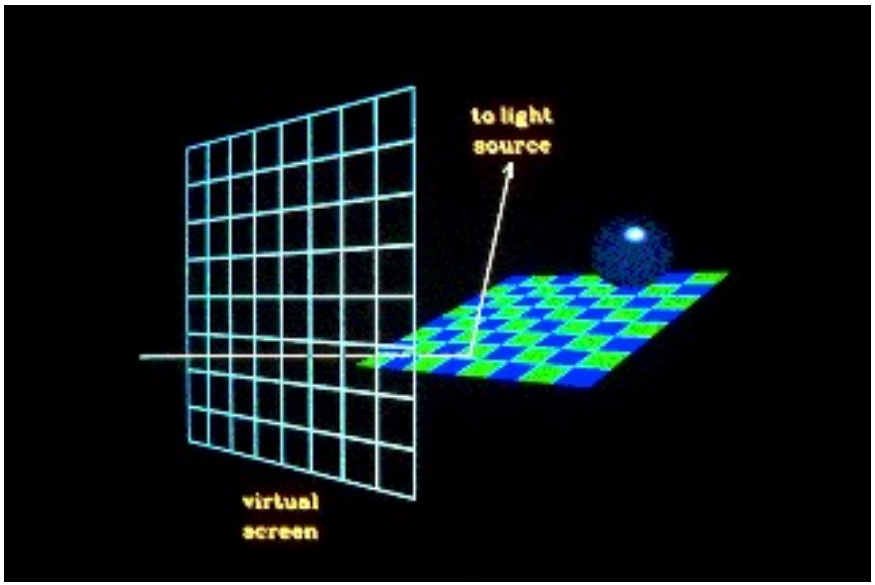


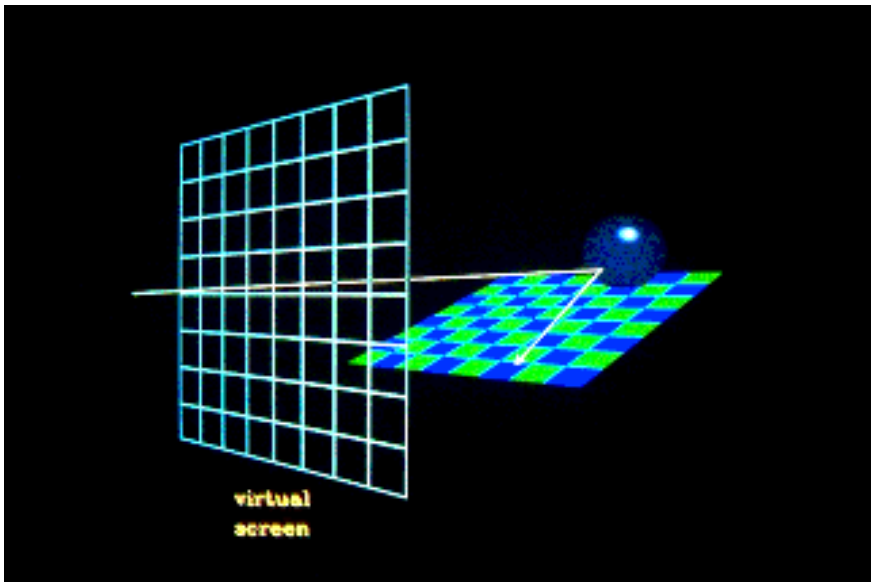
Source: Intel

Video: Quake Wars Raytraced (2008)

Raytrace Idea

Trace light rays backwards from camera to light source:





Source: Siggraph

- Follow ray through reflections and transparent objects
- If no light source is reached, first point is in shadow
- Computational cost dominated by screen size

Raytrace Characteristics

- Huge working sets containing the whole scene
- Exact working set sizes are data-dependent
- Entire scene is shared among all threads
- Memory bandwidth main issue for good speedups

Large working sets, little communication

Streamcluster Overview

- Computes an approximation for the optimal clustering of a stream of data points
- Machine learning application (Princeton)
- Input is a stream of multidimensional points
- Coarse-granular parallelism, static load-balancing
- Medium-sized working sets of user-determined size

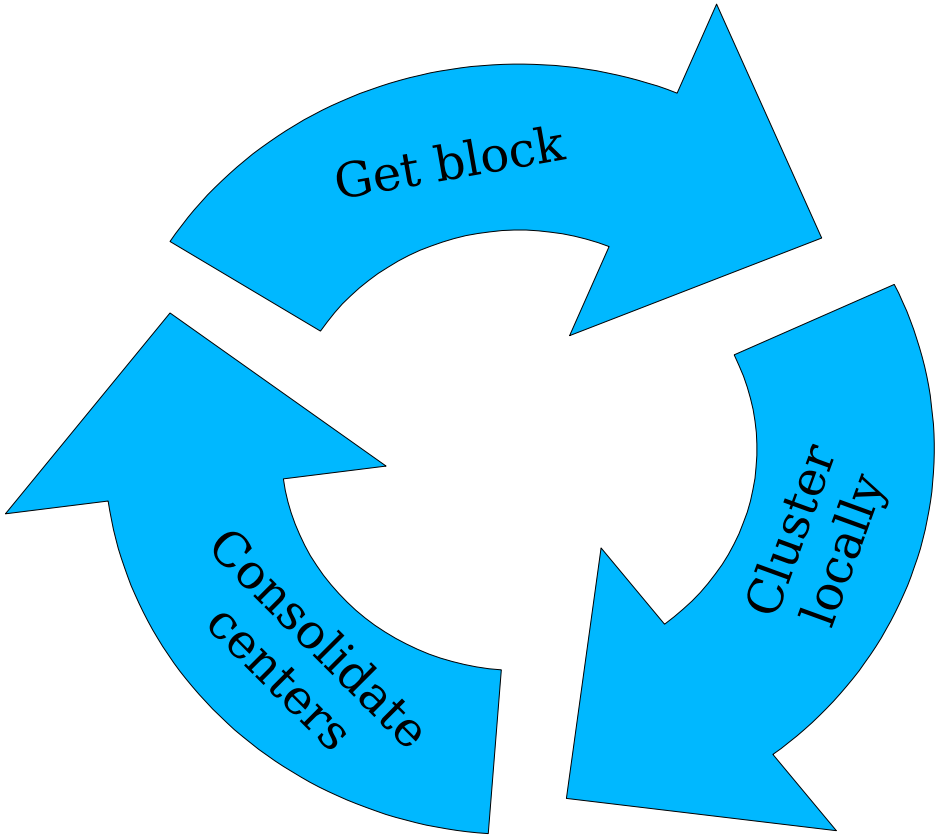
Working set size can be determined at the command line

Streamcluster Rationale

- Clustering is a common problem in many fields like network security or pattern recognition
- Often input data is only available as a data stream, not as a data set (e.g. huge data set that has to be processed under real-time conditions, continuously produced data, etc).
- Approximation algorithms have become a popular choice to handle problems which are intractable otherwise

Streamcluster Idea

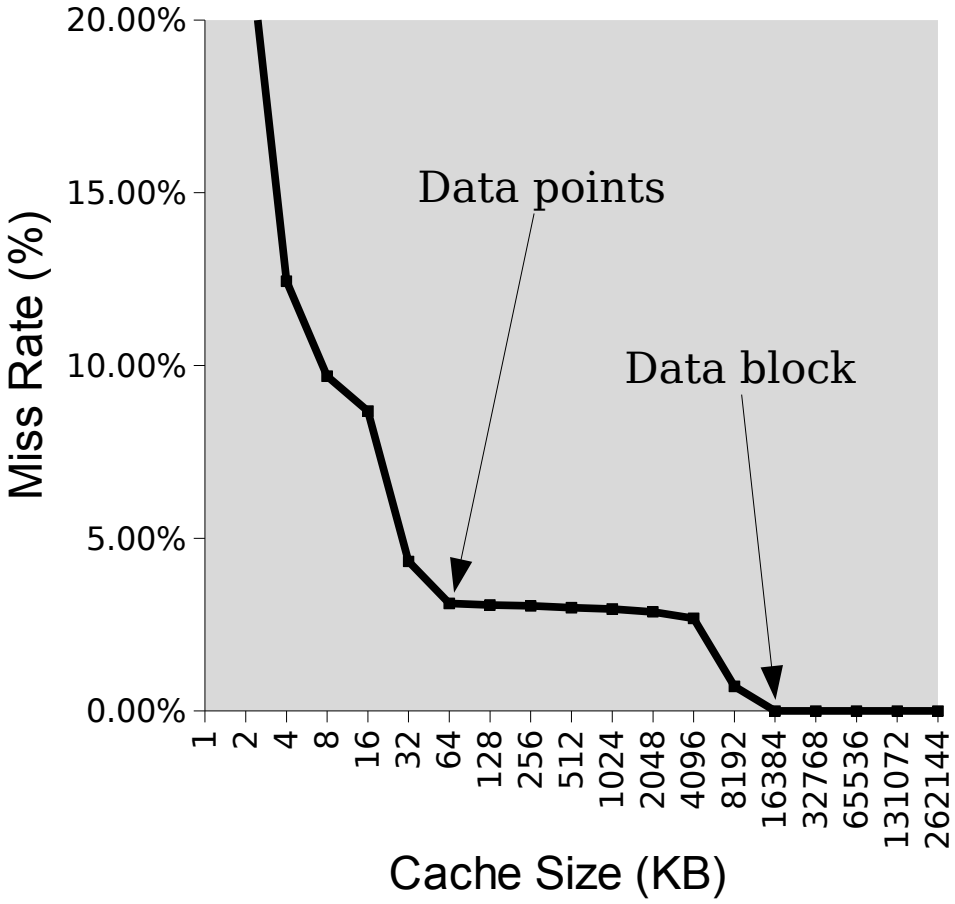
Break stream into blocks, process blocks one by one:



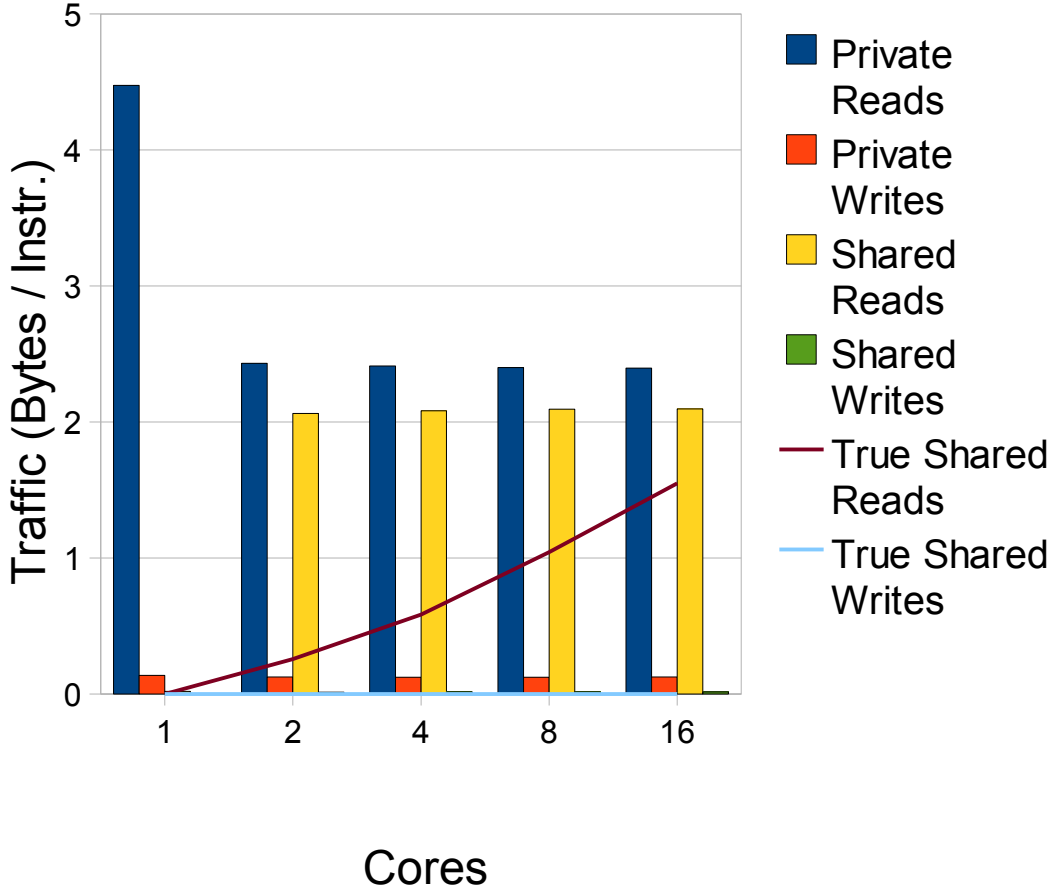
Assign weight to centers according to number of points they represent

Streamcluster Characteristics

Working Sets



Cache Hits



Medium-sized working sets of user-determined size

Swaptions Overview

- Prices a portfolio of swaptions with the Heath-Jarrow-Morton framework
- Computational finance application (Intel)
- Input is a portfolio of derivatives
- Coarse-granular parallelism, static load-balancing
- Medium-sized working sets, little communication

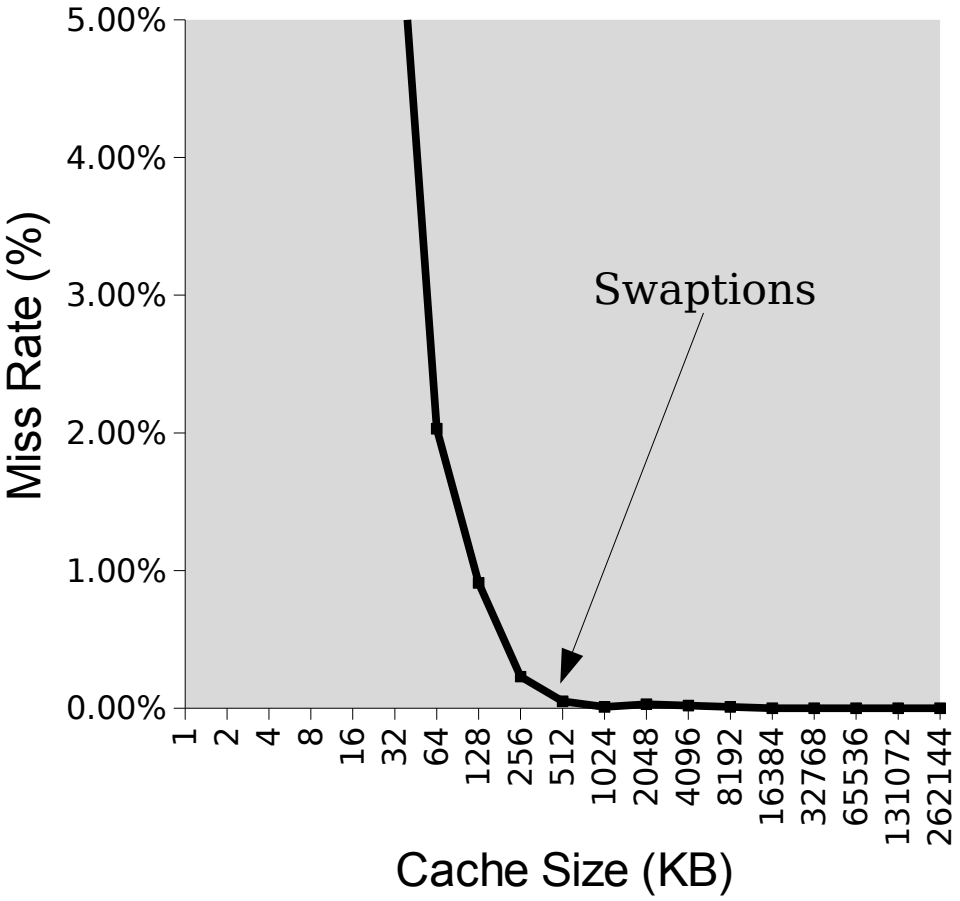
Employs Monte Carlo simulation

Swaptions Rationale

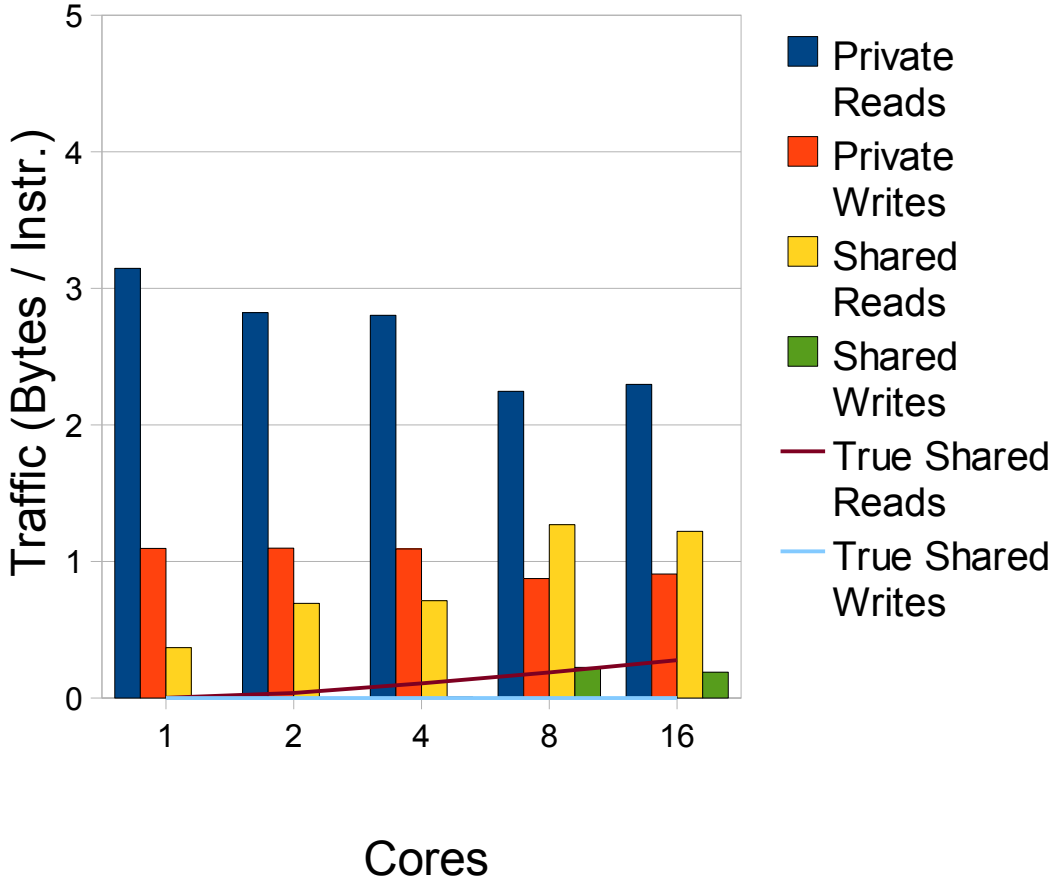
- Computerized trading of derivatives has become wide-spread
- High demand for performance: Saving few milliseconds can earn lots of money
- Monte Carlo simulation is a common approach in many different fields

Swaptions Characteristics

Working Sets



Cache Hits



Medium-sized working sets, little communication

Vips Overview

- Applies a series of transformations to an image
- Media application (Princeton + National Gallery of London)
- Input is an uncompressed image
- Medium-granular parallelism, dynamic load-balancing
- Medium-sized working sets, some sharing

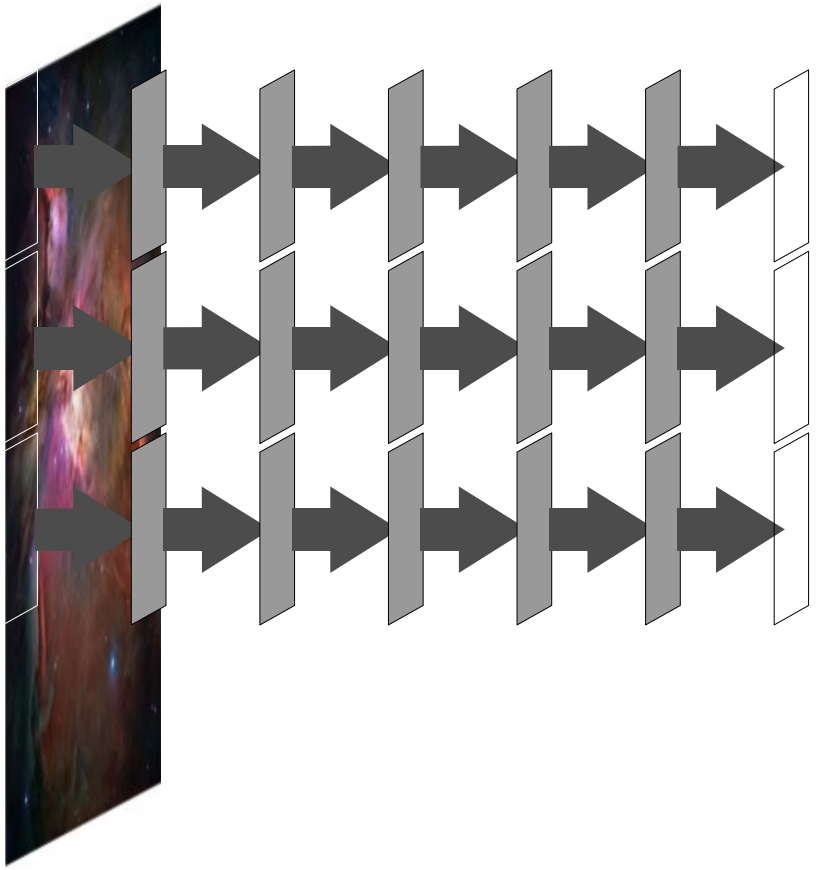
http://www.vips.ecs.soton.ac.uk/

Vips Rationale

- Image processing is one of most common operations for desktops and workstations
- Amount of digital photos grows exponentially
- Professional images can become huge but still need to be handled quickly
- Benchmark based on real print-on-demand service at National Gallery of London

The native input set for vips is a picture of the Orion galaxy with 18,000 x 18,000 pixels.

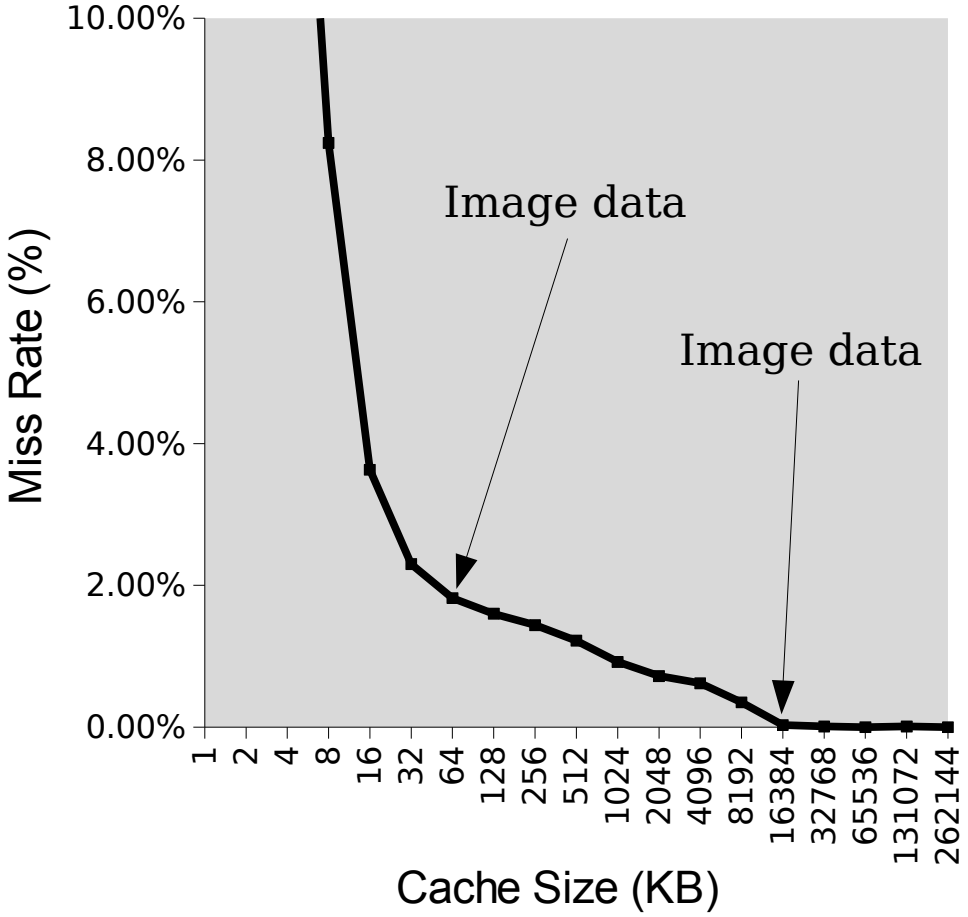
Vips Idea



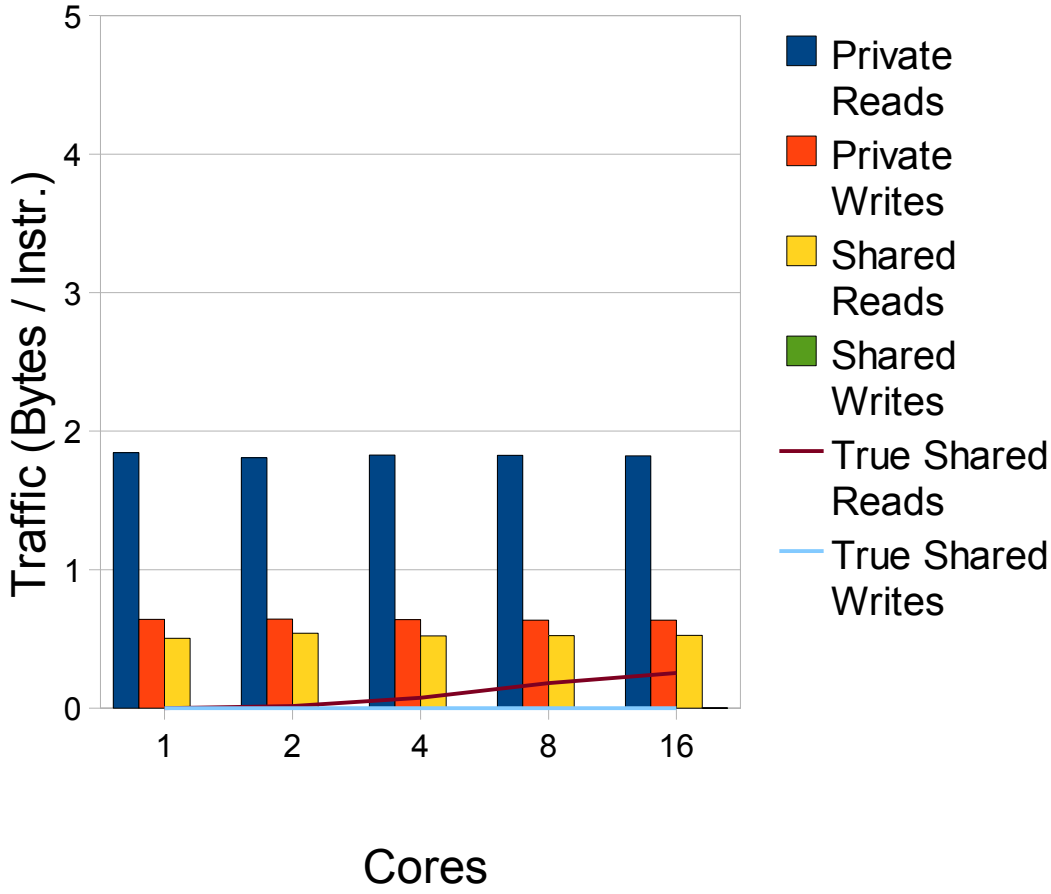
- Fuse all image operations together to a single pipeline
- Replicate pipeline to process image concurrently
- Threading mechanism is transparent to user of vips library

Vips Characteristics

Working Sets



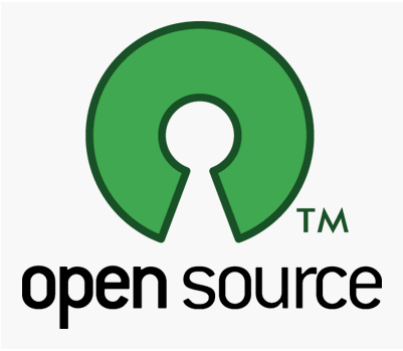
Cache Hits



Medium-sized working sets, some sharing

X264 Overview

- MPEG-4 AVC / H.264 video encoder
- Media application (Princeton + Open Source Community)
- Input is a sequence of uncompressed image
- Coarse-granular pipeline parallelism
- Medium-sized working sets, very communication intensive



http://www.videolan.org/developers/x264.html

X264 Rationale

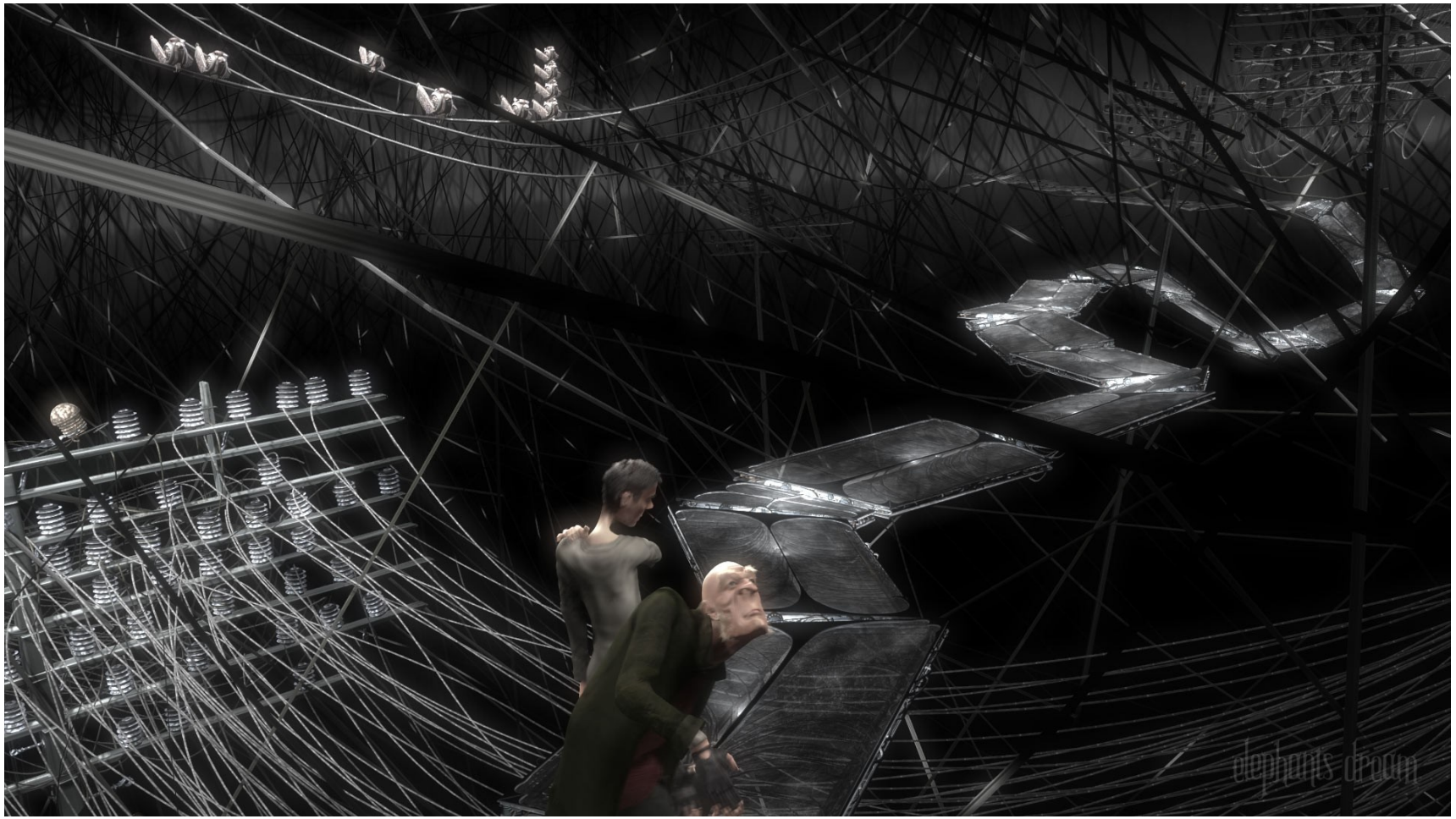
- Increasing storage and network capacity have made videos popular

The input frames for x264 were taken from the open source movie "Elephants Dream" (2006).

- Shift towards digital TV
- MPEG-4 AVC / H.264 is the standard for next-generation video compression

More processing power enables better compression quality

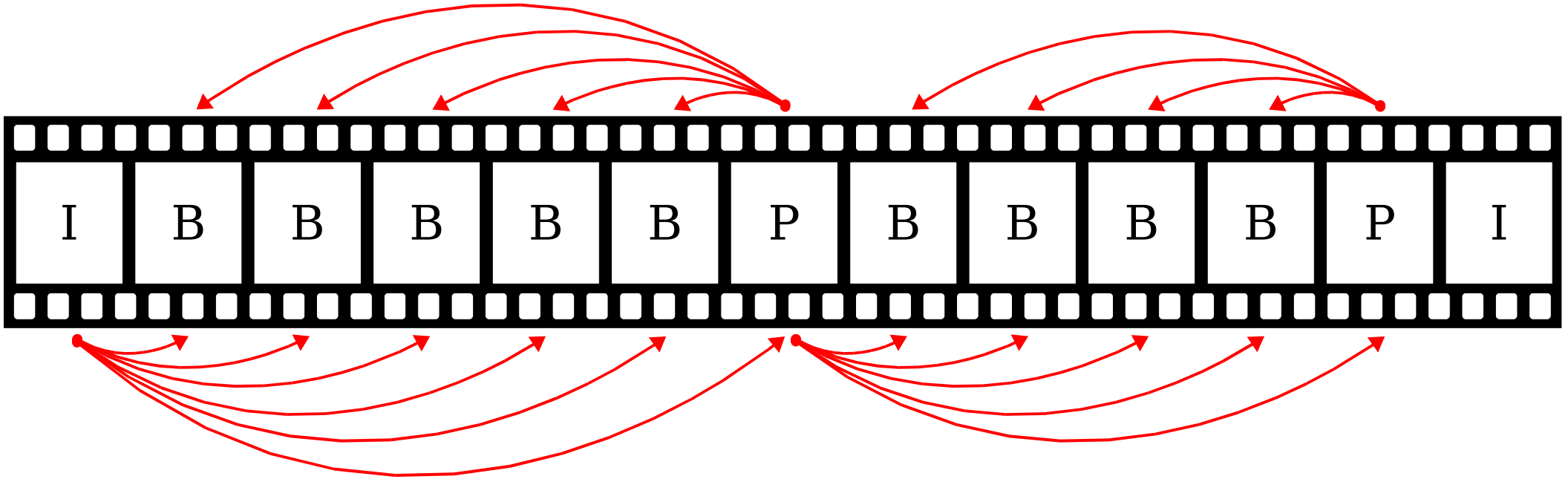
X264 Demonstration



Video: Bridge Scene of Elephants Dream

X264 Idea

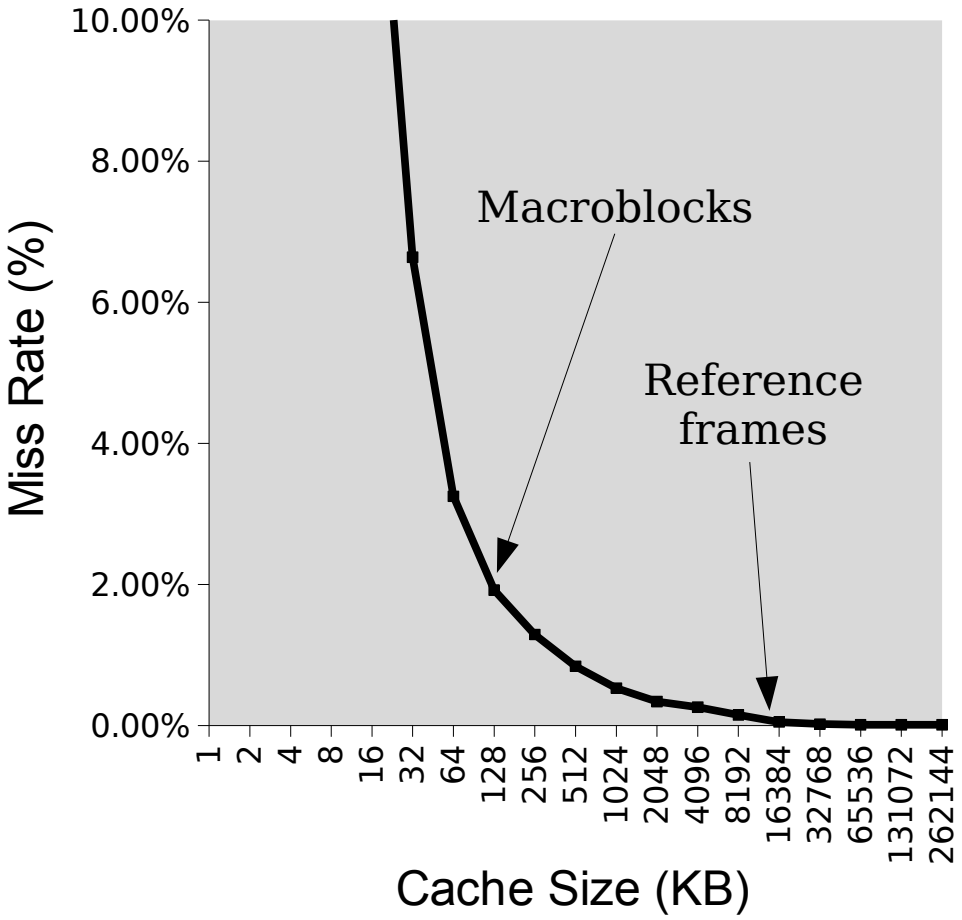
X264 builds a dynamic pipeline shaped like a DAG:



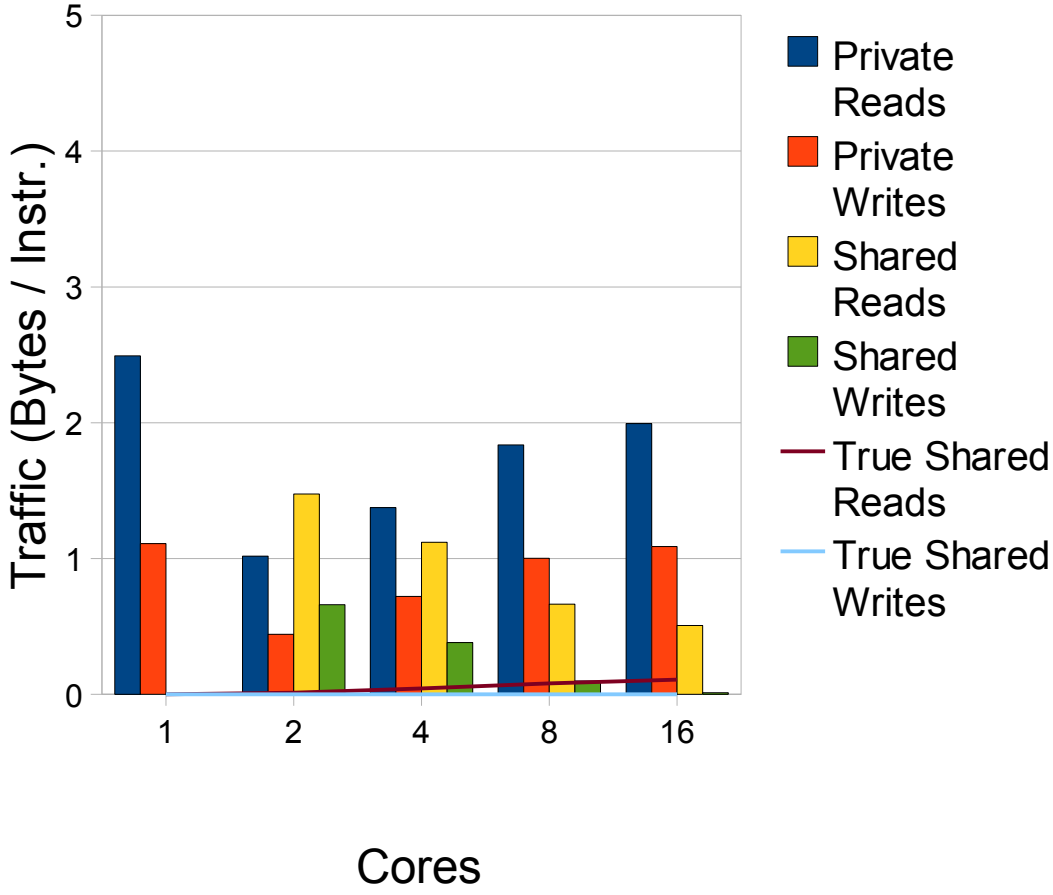
- I-Frames: Self-contained compression
- P-Frames: Dependent on another frame
- B-Frames: Dependent on two other frames

X264 Characteristics

Working Sets



Cache Hits



Medium-sized working sets, very communication intensive

Part 2

Working with PARSEC

Hello World (1)

Run the following command:

```
parsecmgmt -a status
```

Hello World (2)

Run the following command:

```
parsecmgmt -a status
```

You should see some information similar to the following one:

```
[PARSEC] Information about this PARSEC distribution:
[PARSEC]   Version: 2.0

[PARSEC]   Available packages:
[PARSEC]     - blackscholes (apps)
...
[PARSEC]     - yasm (tools)

[PARSEC]   Available inputs:
[PARSEC]     test - Input to test basic program functionality
...
[PARSEC]     native - Huge input for performance analysis on real machines

[PARSEC]   Supported configurations:
[PARSEC]     gcc gcc-hooks gcc-openmp gcc-pthreads gcc-tbb gcc-serial icc
```

parsecmgmt

- 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:

```
parsecmgmt -h
```

Building Workloads

- You can build a PARSEC workload as follows:

```
parsecmgmt -a build -p [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

Building Workloads Quiz

Q: How do you build workload `canneal`?

Building Workloads Answer

Quiz

Q: How do you build package `canneal`?

A: You can use the following command:

```
> 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 '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':
...
```

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 all
```

- Possible aliases are `kernels`, `apps`, `bench`, `libs`, `tools` and `all`
- PARSEC 2.0 also defines three aliases for workloads implementing a specific parallelization model: `openmp`, `pthreads` and `tbb`

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`
- New PARSEC 2.0 build configurations to enable specific parallelizations are `gcc-openmp`, `gcc-pthreads` and `gcc-tbb`

Build Configurations Quiz

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

Build Configurations Answer

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

A: You can use 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:

```
[ARCHITECTURE]-[OSNAME].[BUILDCONF]
```

- You can override this string by defining environment variable `PARSECPLAT`
- PARSEC 2.0 also allows you to append an extension to further distinguish builds

Show Available Installations

- You can see a list of all installed builds if you run:

```
parsecmgmt -a status
```

- `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
...
```

Cleanup

- 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
```

Running Benchmarks

- You can run a PARSEC benchmark as follows:

```
parsecmgmt -a run -p [PACKAGE] -c [BUILDCONF]
           -i [INPUT] -n [THREADS]
```

- 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`

Input 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

Running Benchmarks Quiz

Q: How do you run the serial version of workload `canneal` with input `simsmall`?

Running Benchmarks Answer

Q: How do you run the serial version of workload `canneal` with input `simsmall`?

A: You can use the following command:

```
> parsecmgmt -r 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.
```

Submitting Runs

- You can ask `parsecmgmt` to pass the benchmark program with all arguments to another executable with flag `'-s [EXECUTABLE]'`

- Use this feature to submit workloads to a batch system, start it in an emulator, etc.

- By default, `parsecmgmt` uses `time` in order to measure the execution time of a run

Results obtained with `time` are inaccurate because they include initialization and cleanup time.

Alternate Run Directory

- By default `parsecmgmt` runs all benchmarks inside the PARSEC directory tree.
- This will not work if you cannot write to the directory
- With flag `'-d [RUNDIR]'` you can specify a different directory root for the working directories
- This allows you to keep the PARSEC distribution with all binaries on a read-only network drive or DVD-ROM
- Can have multiple benchmark runs at the same if the run directories are different

Log Files

- `Parsecmgmt` stores all output of builds and runs in log files
- All log files are kept in the `log/` directory of the framework
- Naming convention:

`build_[DATE]_[TIMESTAMP].log`

and

`run_[DATE]_[TIMESTAMP].log`

Documentation

- Comprehensive documentation shipped with PARSEC 2.0
- Full set of man pages available in the `man/` directory
- Add it to the `MANPATH` environment variable to access it (example assumes bash shell):

```
MANPATH=${MANPATH}:${PARSECDIR}/man
```

- We provide a script `env.sh` which does that for you (see next slide)
- Then you can start browsing the documentation as follows:

```
man parsec
```

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

Part 3

Adapting PARSEC

Framework Directory Structure

- PARSEC is composed of the framework and packages

```
parsec-2.0/
   bin/
   config/
   log/
   man/
   pkgs/
      apps/
         ...
      kernels/
         ...
      libs/
         ...
      tools/
         ...
      [GROUPNAME]/
```

Framework executable files

Global configuration files

Log files of builds and runs

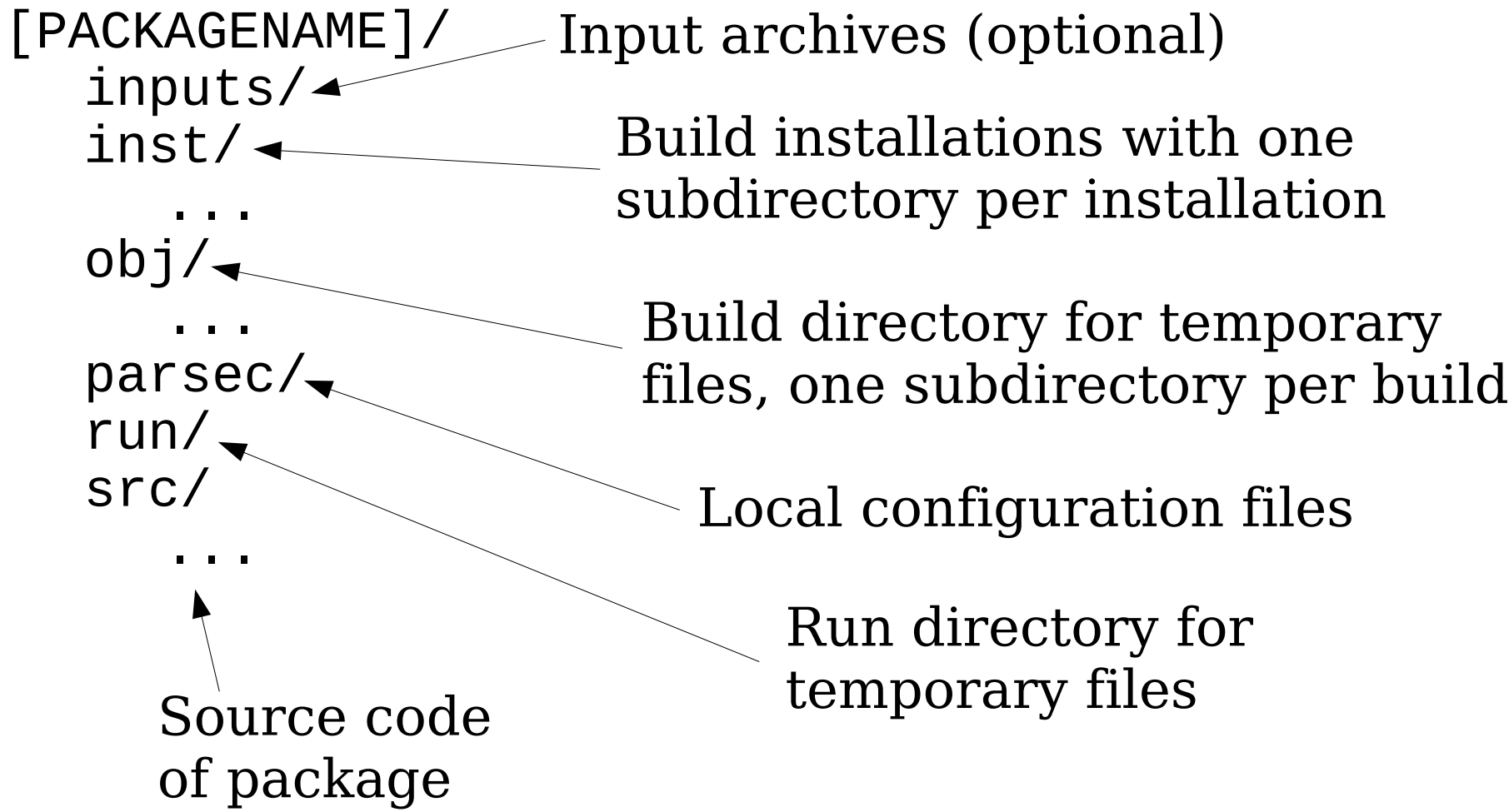
Man pages for PARSEC

One directory for each package group (currently four)

Each group directory contains one directory per package in that group

Package Directory Structure

- Each package directory is structured as follows:



Configuration Files

- Global configuration files (in `config/` directory of framework):
  - PARSEC main configuration file: `parsec.conf`
  - System configurations: `[OSNAME].sysconf`
  - Global build configurations: `[BUILDCONF].bldconf`
  - Global run configurations: `[INPUT].runconf`
- Local configuration files (in `parsec/` directory of each package):
  - Local build configurations: `[BUILDCONF].bldconf`
  - Local run configurations: `[INPUT].runconf`

Managing Build Configurations

- Create a new build configuration:

```
bldconfadd -n [NAME]
```

- In most cases you will want to create a copy of an existing build configuration
- Use flag '`-c`' for a hard copy and flag '`-s`' for a soft copy
- Delete a build configuration:

```
bldconfdel -n [NAME]
```

- Use flag '`-h`' with both tools to get more detailed usage information

Modifying Build Configurations

- You should adapt build configurations to your needs
- Each build configuration has to define:
  - Default environment variables for makefiles (CC, CXX, CFLAGS, ...)
  - Build tool version numbers (CC_ver, CXX_ver, ...)
  - It should define macro `PARSEC_VERSION`
- The global configuration files define all parameters, the local ones adapt them and add additional variables as needed by each package

Build Configuration Quiz

Q: Create a new build configuration `gcc-debug` based on `gcc` that compiles all packages without optimization but with debugging support. Test it on workload `canneal`.

Build Configuration Answer

Q: Create a new build configuration `gcc-debug` based on `gcc` that compiles all packages without optimization but with debugging support. Test it on workload `canneal`.

A: First, create a copy of build configuration '`gcc`':

```
configadd -n gcc-debug -s gcc
```

Next, edit `gcc-debug.bldconfig` in directory `config/` to use the new flags:

```
#!/bin/bash

source ${PARSECDIR}/config/gcc.bldconf

CFLAGS="${CFLAGS} -O0 -g”
CXXFLAGS=”${CXXFLAGS} -O0 -g”
```

Build Information

- `Parsecmgmt` creates a special file '`build-info`' with information about the build in each build installation directory

- File contains details about build configuration and environment at the time of compilation:
  - Exact location and version of all compilers
  - Compiler flags specified by build configuration
  - Modifications of environment variables

- Makes it a lot easier to figure out what was going on if build configurations were modified

Build Information Quiz

Q: How did `parsecmgmt` modify the environment to build the serial version of workload `canneal`?

Build Information Answer

Q: How did `parsecmgmt` modify the environment to build the serial version of workload `canneal`?

A: It's in `build-info` for the `gcc-serial` configuration:

```
PARSEC Compile Information
==========================
Package 'canneal'
Built on Wed May 7 20:24:59 EDT 2008

Configure arguments:  --prefix=/home/cbienia/parsec/parsec-2.0/pkgs/
kernels/canneal/inst/x86_64-linux-linux.gcc-serial

Environment modifications: version=serial

CC: /usr/bin/gcc
Version: gcc (GCC) 4.1.2 20070626 (Red Hat 4.1.2-14)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURP
CFLAGS: -O3 -funroll-loops -fprefetch-loop-arrays -DPARSEC_VERSION=2.0
...
```

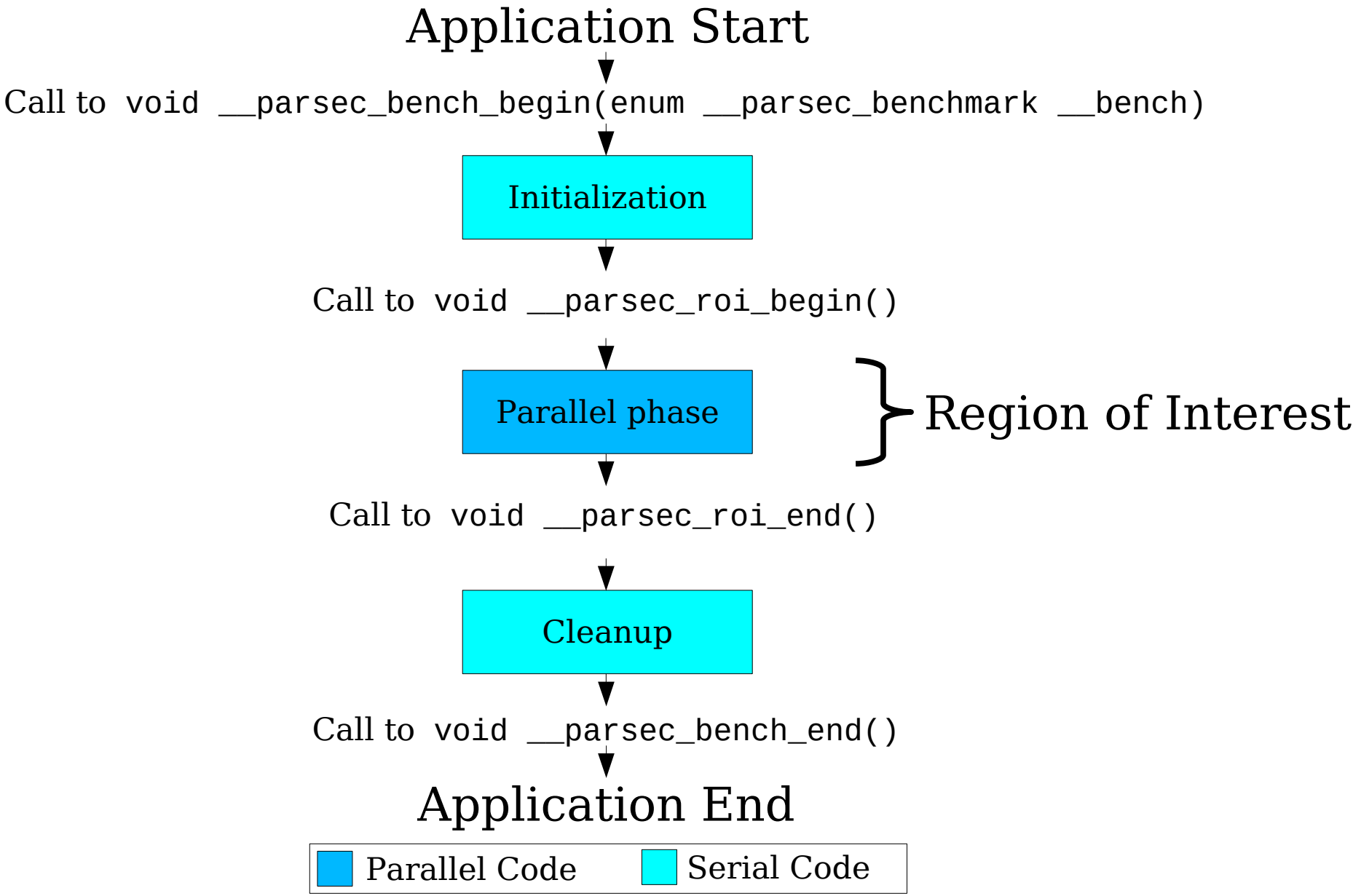
PARSEC Hooks

- Write code once, automatically insert into all workloads simply by rebuilding them
- The hooks API functions are called at specific, predefined locations by all workloads
- Implemented as a library
- Comes with several useful features already implemented (see `config.h` in hooks package)
- Read the man pages for detailed explanations 

Enabling PARSEC Hooks

- Define macro ENABLE_PARSEC_HOOKS (and tell the compiler and linker to use the hooks header files and library)
- The following flags work with gcc:
  - For CFLAGS: `-DENABLE_PARSEC_HOOKS -I${PARSECDIR}/pkgs/hooks/inst/${PARSECPLAT}/include`
  - For LDFLAGS: `-L${PARSECDIR}/pkgs/libs/hooks/inst/${PARSECPLAT}/lib`
  - For LIBS: `-lhooks`
- The build configuration gcc-hooks does this already by default

PARSEC Hooks API



PARSEC Hooks Features

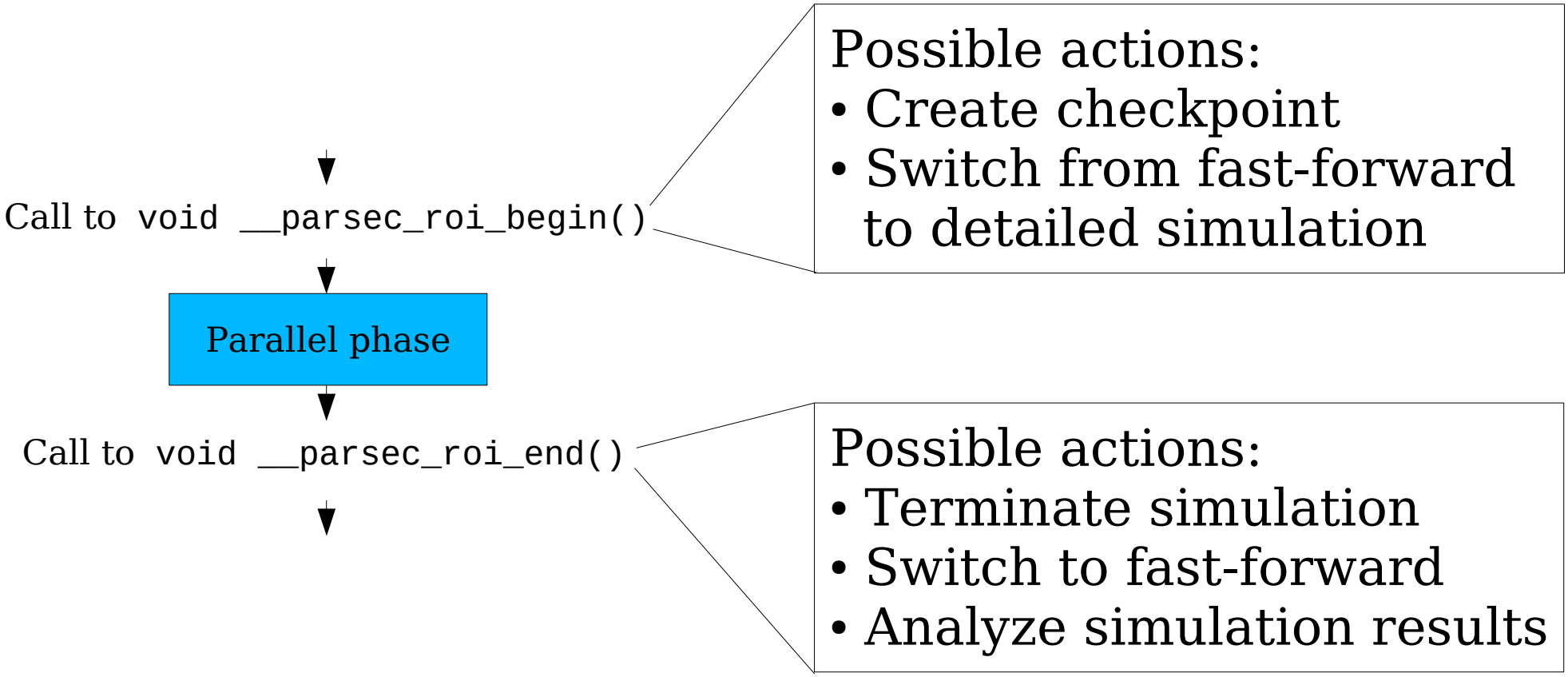
- Measure execution time of ROI
  Define `ENABLE_TIMING` in `config.h` (enabled by default)

- Control thread affinity via environment variables
  Define `ENABLE_SETAFFINITY` in `config.h` (enabled by default, Linux only)

- Execute Simics “Magic Instruction” before and after ROI
  Define `ENABLE_SIMICS_MAGIC` in `config.h` (disabled by default, Simics simulations only)

Assisting Simulations with PARSEC Hooks

You can use PARSEC Hooks to eliminate unnecessary simulation time:



PARSEC Hooks Quiz

Q: Use PARSEC hooks to print out “Entering ROI” if build configuration `gcc-debug` is used. Test it with `canneal`.

PARSEC Hooks Answer (1)

Q: Use PARSEC hooks to print out “I like PARSEC” if build configuration `gcc-debug` is used. Test it with `canneal`.

A: Add a print statement to `__parsec_roi_begin()`:

```
#ifdef ENABLE_MY_OUTPUT
printf(HOOKS_PREFIX“ I like PARSEC\n”);
#endif //ENABLE_MY_OUTPUT
```

Define macro for build configuration `gcc-debug`:

```
#!/bin/bash

source ${PARSECDIR}/config/gcc.bldconf

CFLAGS="${CFLAGS} -O0 -g -DENABLE_MY_OUTPUT”
CXXFLAGS=”${CXXFLAGS} -O0 -g -DENABLE_MY_OUTPUT”
```

PARSEC Hooks Answer (1)

Q: Use PARSEC hooks to print out "I like PARSEC" if build configuration `gcc-debug` is used. Test it with `canneal`.

A: Remove any existing installations of `gcc-debug`:

```
parsecmgmt -a uninstall -c gcc-debug
  -p hooks canneal
```

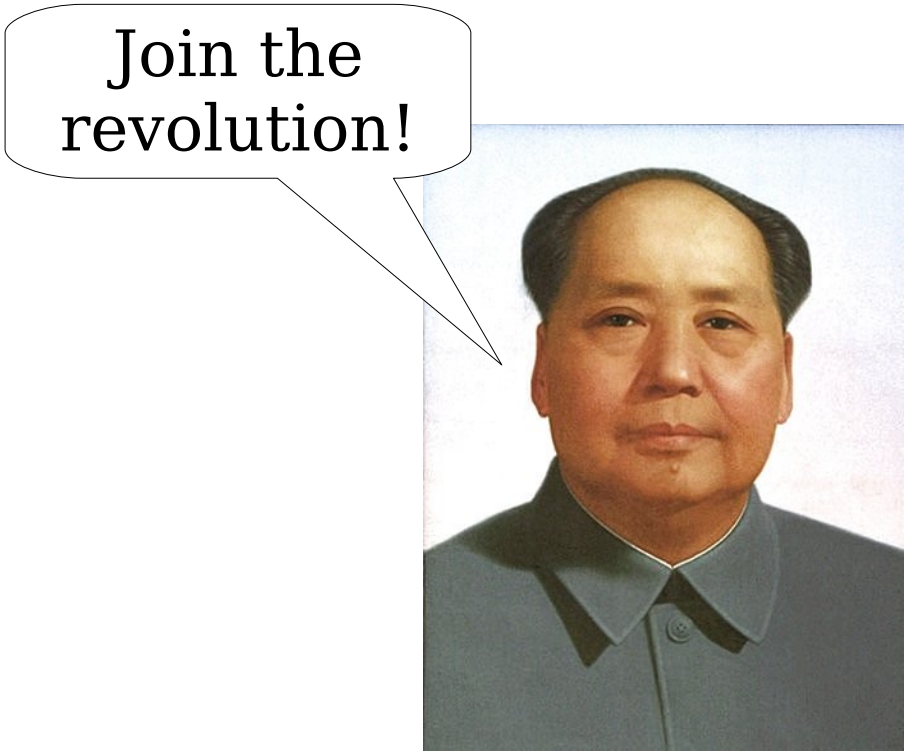
Build and run `canneal`:

```
parsecmgmt -a build -c gcc-debug -p canneal
parsecmgmt -a run -c gcc-debug -p canneal
```

Part 4

Concluding Remarks

PARSEC 3.0



- Release planned for fall 2009
- Several new workloads
- Many patches & bugfixes

We are looking for contributions

References

[1] Christian Bienia and Sanjeev Kumar and Jaswinder Pal Singh and Kai Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. Technical Report TR-811-08, Princeton University, January 2008

[2] Christian Bienia and Sanjeev Kumar and Kai Li. PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors. Technical Report TR-818-08, Princeton University, March 2008

Open Discussion

Where do you think PARSEC should go?

What has to change?

Questions?