ΕΠΛ 605: Προχωρημένη Αρχιτεκτονική Υπολογιστών

PARSEC 3.0

Princeton Application Repository for Shared-Memory Computers (PARSEC)

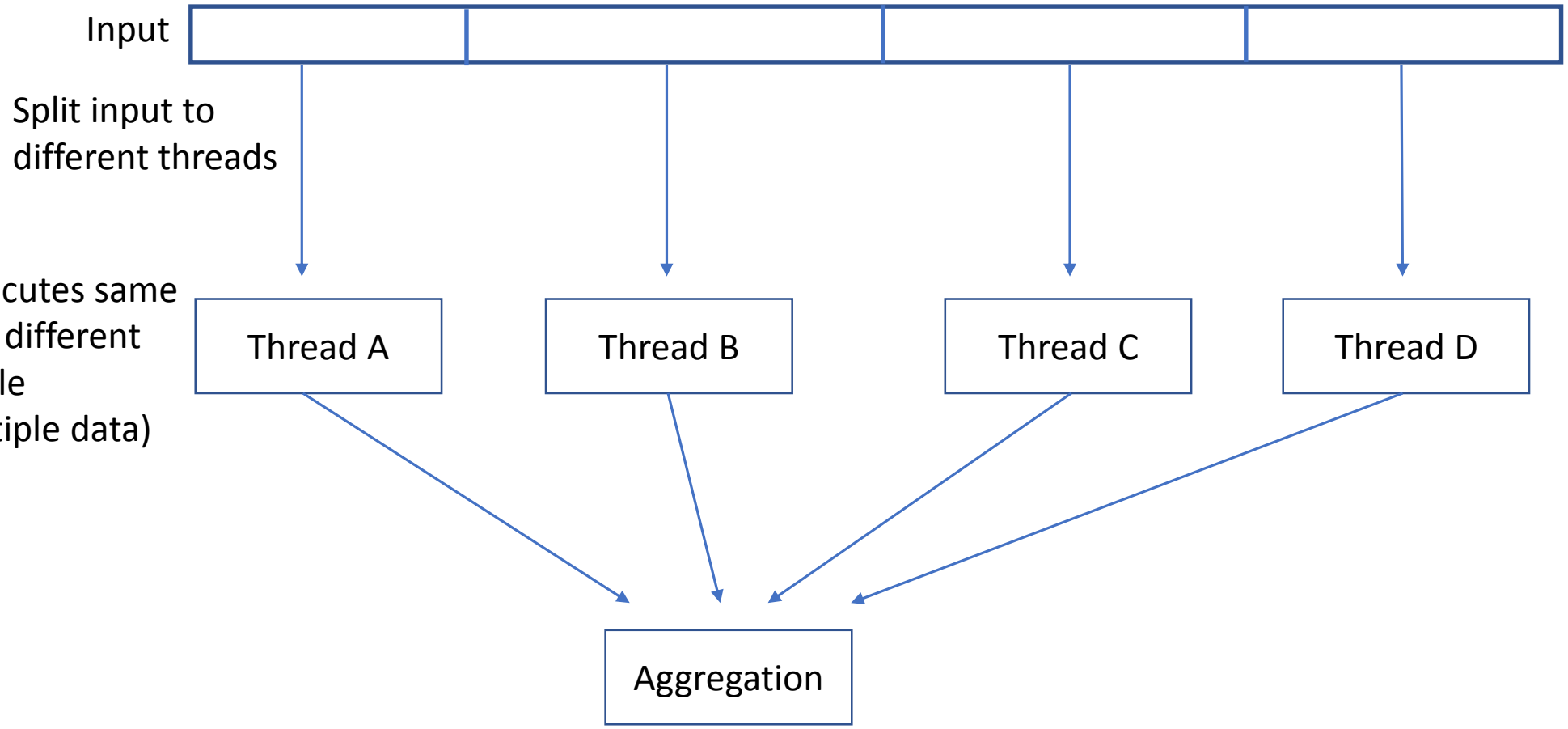
PARSEC suite

- 10 apps : blackscholes bodytrack facesim ferret fluidanimate freqmine raytrace swaptions vips x264
- 3 kernels: canneal dedup streamcluster
- Characteristics
  - Represent emerging workloads such as face recognition, data mining and computer vision
  - Parallel benchmarks that launch multiple threads and utilize multiple cores
  - Shared memory accesses -> stress the cache coherency subsystem
- Download link : http://parsec.cs.princeton.edu/download/3.0/parsec-3.0.tar.gz

Workloads

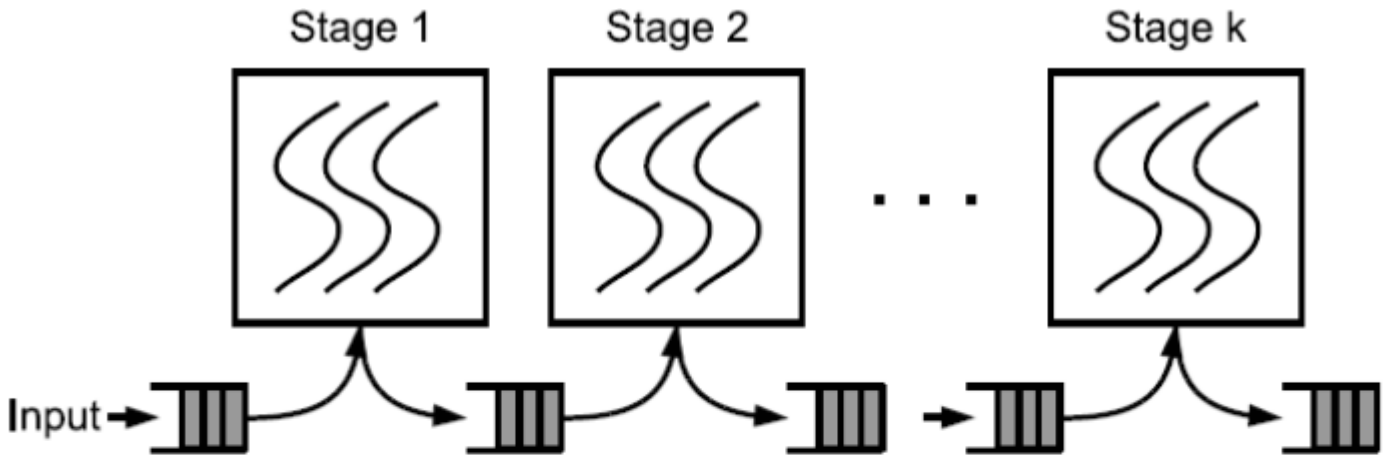
Program	Application Domain	Parallelization		Working Set	Data Usage	
		Model	Granularity		Sharing	Exchange
blackscholes	Financial Analysis	data-parallel	coarse	small	low	low
bodytrack	Computer Vision	data-parallel	medium	medium	high	medium
canneal	Engineering	unstructured	fine	unbounded	high	high
dedup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

Data parallelism



Task parallelism

- Each thread has a different role (executes different function)
  - MIMD (multiple input multiple data) model
- Pipelining parallelism is a form of task parallelism



Parsec 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

Parsec distribution contents

cd parsec-3.0/; ls *

config:

gcc.bldconf gcc-pthreads.bldconf gcc-openmp.bldconf gcc-serial.bldconf gradarwin.sysconf gcc-hooks.bldconf gcc-gcc-tbb.bldconf icc

bin:

parsecmgmt bldconfadd bldconfdel Install

pkgs:

apps kernels libs netapps tools

config/gcc.bldconf #global configuration file with the global gcc flags
config/ gcc-pthreads.bldconf #for compiling benchmarks with pthreads
config/ gcc-openmp.bldconf #for compiling benchmarks with openmp

To add a custom configuration do ./bin/`bldconfadd -n gcc-pthreads-custom -c gcc-pthreads`
Then you can edit the custom configuration vi ./config/pthreads-custom.bldconf

To build and run benchmarks use ./bin/parsecmgmt

Benchmark binaries are saved under /pkgs

Build benchmarks

Build openmp version of all parsec benchmarks

```
bin/parsecmgmt -a build -p parsec -c gcc-openmp
```

Build openmp version of the canneal benchmark

```
bin/parsecmgmt -a build -p parsec.canneal -c gcc-openmp
```

Running benchmarks

parsecmgmt -a run -p parsec.canneal -n 2 -c gcc-pthreads -i native -k

#runs canneal pthread version with 2 threads (-n) with native inputs (-i)

#-k is for keep & use run directory as found, do not unpack inputs for benchmark execution. Assume #everything is already set up.

To retrieve execution time read the pkgs/kernel/canneal/run/benchmark.out file

Example of script to run parsec benchmarks

```
#!/bin/bash
PARSECMGMT=/home/zhadji01/parsec-3.0/bin/parsecmgmt
PKGS_DIR=/home/zhadji01/parsec-3.0/pkgs

if [[ -z $1 || -z $2 ]]; then
        echo "please give bechmark name and number of threads"
        echo "e.g. run_parsec.sh x264 4"
        exit
fi

name=$1
threads=$2

###bodytrack, blacksholes only support openmp

conf=gcc-pthreads
if [ $name = "freqmine" ]; then
        conf=gcc-openmp
        export OMP_PROC_BIND=TRUE
fi

type="apps"
if [[ $name = "canneal" || $name = "dedup" || $name = "streamcluster" ]]; then
        type="kernels"
fi

if [[ ! -d "$PKGS_DIR"/"$type"/"$name"/run ]]; then
        $PARSECMGMT -a run -p parsec."$name" -n "$threads" -c "$conf" -i native
else
        $PARSECMGMT -a run -p parsec."$name" -n "$threads" -c "$conf" -i native -k
fi

sleep 2; #wait a bit.. make sure everything is written down

exec_time_in_minutes=`cat /home/zhadji01/parsec-3.0/pkgs/"$type"/"$name"/run/benchmark.out | grep real | tail -n 1 | awk '{print $2}' | tr 'm' ' ' | tr 's' ' ' | tr '.' ' ' | awk '{minutes=$1+$2/60+$3/60000; print minutes}'`
exec_time_in_seconds=`cat /home/zhadji01/parsec-3.0/pkgs/"$type"/"$name"/run/benchmark.out | grep real | tail -n 1 | awk '{print $2}' | tr 'm' ' ' | tr 's' ' ' | tr '.' ' ' | awk '{seconds=$1*60+$2+$3/1000; print seconds}'`

echo "$name executionTimeInMinutes: $exec_time_in_minutes"
echo "$name executionTimeInSeconds: $exec_time_in_seconds"
```

More Information

- http://parsec.cs.princeton.edu/download/tutorial/3.0/parsec-tutorial.pdf