FINAL EXAM SCHEDULER

4005-735-01 Parallel Computing I

G2. Team Kyz

Kevin Cheek, Yandong Wang, Ziyan Zhou

http://tinyurl.com/ritkyz

FINAL EXAM SCHEDULING PROBLEM

- Sections

0101-301-02	FINANCIAL ACCOUNTING	KEARNS,F	Open	39	38	TR	400P	55	
0101-301-03	FINANCIAL ACCOUNTING	KEARNS,F	Close	40	40	TR	1200N	15	
0101-301-71	FINANCIAL ACCOUNTING	EVANS,W	Open	40	38	T	600P	950P	12
0101-301-90	FINANCIAL ACCOUNTING	LEBOWITZ,P	Close	25	25	NA	ONLINE	CO	
0101-302-01	MANAGEMENT ACCOUNTING	OLIVER,B	Open	40	39	TR	200P	35	
0101-302-02	MANAGEMENT ACCOUNTING	DEY,R	*Open	40	38	MW	800A	950A	12
0101-345-71	ACCOUNTING INFO SYSTEMS	NEELY,M	*Open	28	17	W	600P	950P	12
0101-409-01	FINAN. REPT. & ANLYS. II	KEARNS,F	*Open	40	26	MW	12		
0101-494-71	COST ACCTG TECH ORG	MORSE,W	*Open	15	9	W	600P	950P	12
0101-523-90	ADVANCED TAXATION	KLEIN,R	Open	25	19	NA	ONLINE	COURSE	NA
0101-540-71	ADVANCED ACCOUNTING	OLIVER,B	*Open	20	12	M	600P	95	
0101-554-01	FORENSIC&FRAUD ACCOUNTNG	KLEIN,R	Open	12	0	W	400P	55	
0101-703-71	ACCTG FOR DECISION MAKER	MORSE,W	*Open	35	31	T	600P	92	
0101-703-90	ACCTG FOR DECISION MAKER	KLEIN,R	Open	25	23	NA	ONLINE	CO	
0101-704-71	CORP FINANCIAL REPT I	KARIM,K	Open	35	15	W	600P	920P	12
0101-706-71	COST MANAGEMENT	KARIM,K	Open	34	16	T	600P	920P	12	11

- Slots
  - MTWRF 8:00-10:00 10:15-12:15p 12:30p-14:30p 14:45p-16:45p
- Students

Paper #1: Stochastic Search Algorithms for Exam Scheduling

By Mansour and Timany, published in International Journal of Computational Intelligence Research in 2007

- Exam scheduling as a modified weighted graph coloring problem
- Algorithms
  - Simulated Annealing Algorithm (SA)
  - Genetic Algorithm (GA)

Problem Modeling

A modified weighted graph coloring problem:

- vertex = exam
- weight on vertex = # of students taking the exam
- edge joining two vertices = someone is taking both exams
- weight on edge = # of students taking both exam
- # of colors = # of exam slots
- OF1 = ranking function of the schedule considering all factors:
  - Exam conflicts
  - Consecutive exams
  - More than two exams on the same day
  - Room capacity conflicts

Simulated Annealing Algorithm

```
Initial configuration = random;
Initial temperature T(0) = 0.93; //high initial acceptance
Freezing temperature Tf = 2^-30; //uphill moves impossible

while(T(i) > Tf and not converged) {
  repeat (# of slots) * (# of exams) times
   generate_function();
  save_best_so_far(); //smallest OF1
  T(i) = phi * T(i); //phi = 0.95
}

function generate_function() {
  perturb(); //randomly change one exam's slot to another slot
  if (ΔOF1 <= 0)
   accept();
  else if (ramdon() < e^(-ΔOF1/T(i)))
   accept(); //accept with a probability
}
```

Genetic Algorithm

```
Randomly generate initial population size POP;
Evaluate fitness of individuals;

repeat {
  rank individuals and allocate reproduction trials;

  for i=1 to POP step 2 {
   randomly select two parents from list;
   apply crossover and mutation;
  }

  apply hill-climbing to offspring //hybridization

  evaluate fitness of offspring;
  save_best_so_far();

} until converge
```

PAPER #2: PARALLEL GENETIC ALGORITHM TAXONOMY

By Nowostawski and Poli, published in ACM Journal in 1999

- Types of Parallel Algorithms
  - Master-Slave parallelism
    - Synchronous
    - Asynchronous
  - Static subpopulations with migration
  - Static overlapping subpopulations (without migration)
  - Massively parallel genetic algorithms
  - Dynamic demes
  - Parallel steady-state genetic algorithms
  - Parallel messy genetic algorithms
  - Hybrid methods

PAPER #3: UNLOCKING CONCURRENCY

By Tabatabai, Kozyrakis and Saha, published in ACM Journal in 2007

- Introduce a new concept for parallel programming

Traditional Way of Synchronization

- The traditional way is to use lock for synchronization.
  - For example in JAVA, programmer use "synchronize" key word to write exclusive block.
- Pitfall:
  - Simplistic coarse-grained locking does not scale well
  - Sophisticated fine-grained locking introduces the risk of deadlocks and data races.

Transactional Memory Management

- A memory transaction is sequence of memory operations, meaning they are an all-or-nothing sequence of operation
- A memory transaction runs in isolation, meaning it executes as if it is the only operation running on the system

In fact, In our "Parallel Java Library", there exist some objects in "edu.rit.pj.reduction” whose memory operation are like transaction, or in some degree have the same concept.

For example: object: SharedLong, SharedBoolean, SharedByte, and other Shared Objects are all the wrapper classes of the objects in "java.util.concurrent.atomic" as original object.

```
public SharedLong() {
      myValue = new AtomicLong();
  }
```

- There has a description for "java.util.concurrent.atomic" in java official document:
  - "A small toolkit of classes that support lock-free thread-safe programming on single variables."
- Lock-based vs. Transactional Map Data Structure

A

```
class LockBaseMap implements
Map {
  Object mutex;
  Map m;

  LockBaseMap(Map m) {
    this.m = m;
    mutex = new Object();
  }

  public Object get() {
    synchronized(mutex) {
      return m.get();
    }
  }
}
```

B

```
class AtomicMap implements Map
{
  Map m;


  AtomicMap(Map m) {
    this.m = m;
  }


  public Object get() {
    atomic {
      return m.get();
    }
  }
}
```

Comparison with Synchronization

- Atomic can allow concurrent read operations to the same variable. In a parallel program, it is safe to allow two or more threads to read the same variable concurrently
- Basic mutual exclusion locks don't permit concurrent reads, to allow concurrent readers, the programmer has to use special reader-write locks, increasing the program's complexity
- Atomic can allow concurrent read and write operations to different variables
- In synchronize vision, this is a tedious and difficult task, and will introduce some risk of deadlocks and data races
- Transaction also provide failure atomicity
- In lock based code, programmer must manually restore the data.

Data Versioning and Conflict Detection

- Transaction memory transfers the burden of concurrency management from the application programmers to the system design
- As transactions execute, the system must simultaneously manage multiple versions of data
  - eager versioning
  - lazy versioning
- conflict detection and resolution are essential to guarantee atomic execution. detection relies on tracking the read set and write set for each transaction.
  - pessimistic conflict detections
  - optimistic conflict detection

Final Exam Scheduler

- Multiple approaches using different algorithms
  - Random Trial
    - Randomly generates a schedule and hope to hit the best one
  - Brute-Force Algorithm
    - Iterate through all possible solutions and find the best
    - Requires small data set to complete in reasonable time
  - Simulated Annealing Algorithm (SA)
  - Genetic Algorithm (GA)

Brute-Force Sequential Approach

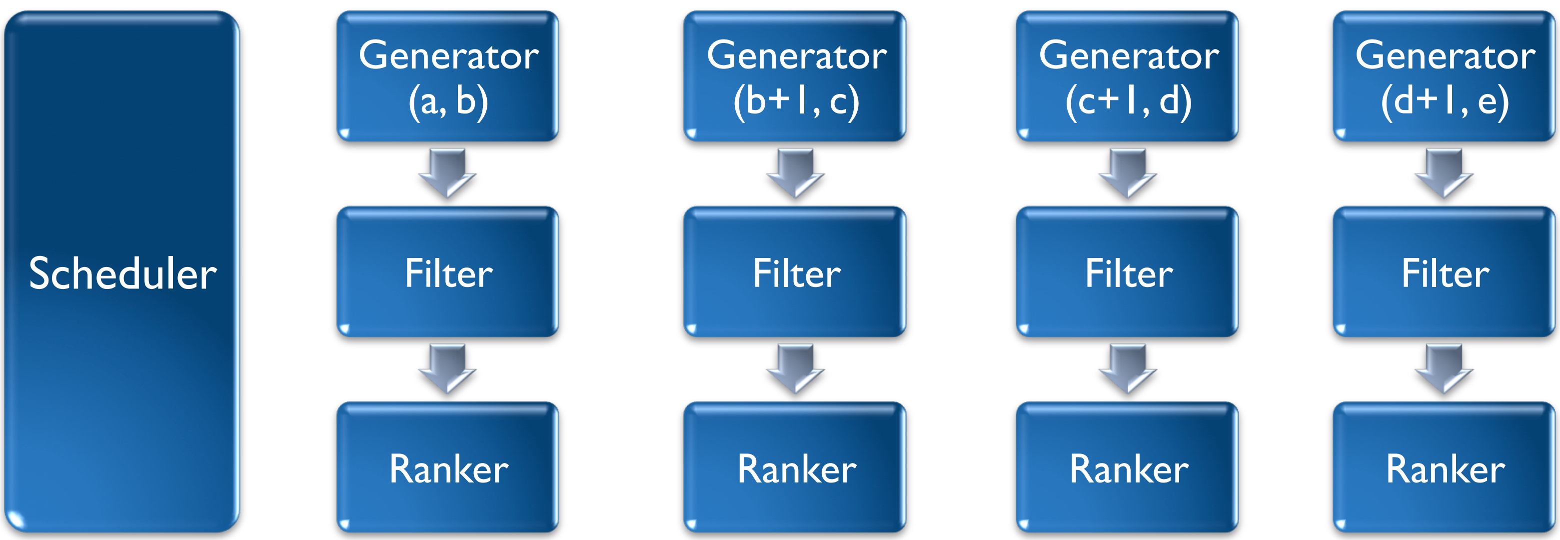

```
Load data from MySQL database;
Create generator, filter, ranker;

while (generator has new schedule) {
  generator.generates a new schedule;
  filter.flags obviously bad schedule;
  ranker.rank(schedule);
  save_best_so_far();
}

function rank(schedule) {
  overall rating of the schedule;
  foreach(student in students) {
    student vote on schedule;
  }
  return average vote + overall;
}
```

Brute-Force Generator

- Brute-Force Schedule Generator takes two schedules (start, end) at construction, will start from one schedule and iterate through all schedule until the end schedule is reached (inclusive).
  - So, given # of time slot = 3, new Generator({Math=>0, CS4=>0}, {Math=>1, CS4=>1}) will generate the following schedules:
    - Math=>0, CS4=>0;
    - Math=>0, CS4=>1;
    - Math=>0, CS4=>2;
    - Math=>1, CS4=>0;
    - Math=>1, CS4=>1;

Brute-Force Parallel Approach (Cluster)

- Agenda Parallelism with Reduction
- Use dynamic scheduler on Master node
  - Schedule 0-1000, 1001-2000, 2001-3000 and so on, each worker grabs new range when done with the current assignment
- Each node keeps a local best, use reduction in the end to gather result



Other Thoughts

- Parallel Genetic Algorithm (GA)
  - Static subpopulations with migration
    - Each node has a local subpopulation
    - Crossover and mutation done in local subpopulation
    - Occasional migration between nodes
    - Master node keeps track of convergence and assign new population
- Simulated Annealing Algorithm (SA)
  - Sequential dependency on temperature
    - Size-up approach: each node has its own annealing process, and reduction in the end
    - Speed-up approach: distribute perturb at each temperature across all nodes, barrier at each temperature

THANK YOU FOR LISTENING

- G2. Team Kyz
  - Kevin Cheek
  - Yandong Wang
  - Ziyan Zhou
- Visit us at: http://tinyurl.com/ritkyz