Finale

Chris Kauffman

Last Updated:
Wed Dec 15 09:43:08 AM CST 2021

Logistics

Schedule

Mon 12/13	Applications Parallel Languages
Wed 12/15	Review A2 Due Course Evals Due
Fri 12/17	A2 Late Deadline
Mon 12/20	Final Exam 1:30-3:30pm Lecture Location

Today

- Review Exercises
- P2 Questions

Further Coursework

- **CSCI 8205 Parallel Computer Organization**: Study hardware issues associated with parallel / multicore machines. Requires significant Hardware background.
- **CSCI 5304 Computational Aspects of Matrix Theory**: Deep dive into using matrices and linear algebra in computing. Essential stuff for those in scientific computing with any self-respect.
- **CSCI 8314 Sparse Matrix Computations**: Focused on sparse operations, offered periodically.
- **Application Areas:** Have seen that machine learning, graphics, cryptography, physical simulations, etc. all benefit from parallel computing. Find a serial algorithm in your domain and parallelize it!

Survey Says ...

SRTs Response Rate

Responded	Invited	%Rate
46	56	82.14%

- Thanks to all that have responded; SRTs stay open until 11:59pm last day of classes
- As promised, Final Exam Question Reveal

Final Exam Question

The following MPI fragment demonstrates a certain communication pattern which is done inefficiently. Reorganize the code and replace inefficient calls with an equivalent and more efficient collective communication pattern.

```
int N, *buf;                              // data for communication
...

if(proc_id == 0){                         // communication
  for(int i=1; i<total_procs; i++){
    MPI_Send(&N,  1, MPI_INT, i, 1, MPI_COMM_WORLD);
    MPI_Send(buf, N, MPI_INT, i, 1, MPI_COMM_WORLD);
  }
}
else{
  MPI_Recv(&N,  1, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  buf = malloc(N * sizeof(int));
  MPI_Recv(buf, N, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}

printf("%2d: N = %d\n",proc_id,N);  // procs print their data
for(int i=0; i<N; i++){
  printf("%2d: buf[%d] = %d\n",proc_id,i,buf[i]);
}
```

Review Topic: GPU Thread Sync

A

We have seen that CUDA provides the `__synchthreads()` function to synchronize threads within a thread block.
Why is this function necessary? Give an example where it would be useful.

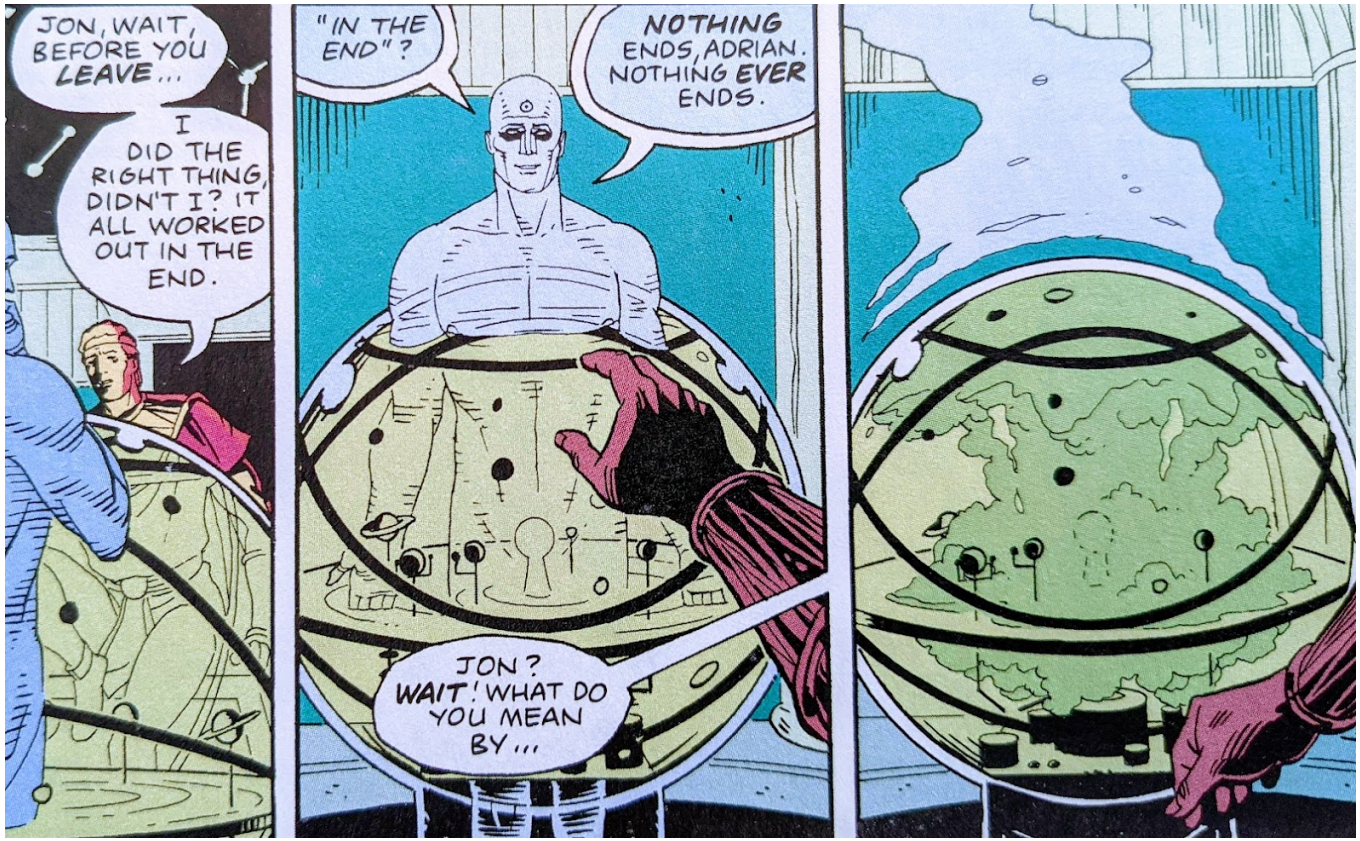
B

What are the limits to `__syncthreads()`?
Which threads does it synchronize and which threads does it not?
What alternatives are available for thread synchronization?

Review Topic Requests

- Distributed Memory Architecture
- Shared Memory Architecture
- MPI for Distributed Memory Programming
- PThreads + OpenMP for Shared Memory Programming
- CUDA for GPU Programming
- Communication Patterns
  - Broadcast, Scatter/Gather, Reductions
  - Synchronization between procs/threads
- Input/Output Problem Decomposition
- Parallel Problems/Applications
  - Sums, Min/Maxes
  - Matrix Multiplication
  - Solving Linear Systems
  - Sorting
  - Basics of Fluid Dynamics, N-Body simulation, Neural Networks, Crypto Mining

Nothing Ever Ends



By now you should realize that what you learned

- Will come up again showing whether you learned it well the first time or need another pass.
- Will change in the future and make you feel old.

Expect this and stay stay patient.

Conclusion

It's been a hell of a semester.
I'm proud of all of you.
Keep up the good work.
Stay safe. Happy Hacking.