15-418/618 FINAL PROJECT CHECKPOINT: CacheM ALL Sim
Cache Memory Access Load Latency Simulator

Original Schedule

During our project proposal, we had planned a schedule that looked something like this:

Week 1: October 29th - November 4th	● Meet Course Instructors Tuesday(10/29, 10:15 am) ● Decide on our project and submit a proposal (10/30, 23:59 hrs) ● Study the starter codebase
Week 2: November 5th - November 11th	● Implement NUMA ● Create a way to calculate clock ticks
Week 3: November 12th- November 18th	● Implement UMA+NUMA architecture
PROJECT CHECKPOINT: November 18th	
Week 4: November 19th - November 25th	● Finish remaining implementations ● Generate memory traces
Week 5: November 26th - December 2nd	● Run traces and analyze performance results
Week 6: December 3rd - December 9th	● Work on the final report and project presentation
FINAL PROJECT DUE: December 9th	
PROJECT POSTER SESSION: December 10th	

Work Done so Far

During week 1, we met with Brian Railing and talked about our various ideas. Out of those ideas we decided to go with a cache coherence simulator, which simulates traces in UMA and SMP architectures and creates a comparative study of the latency and performances. After deciding that, we wrote a project proposal and submitted it on Campus by the 30th of October. On the weekend following we met and documented the simulator and made note of how it worked.

After that for week 2, we started implementing a mechanism to be able to calculate clock ticks in both NUMA and UMA architectures. We laid the following standards:
- If data was found in the local memory: it should take 1 clock cycle to access it
- If data found in other memory: it should take 30 clock cycles.
- If another memory indicates that the home memory does not have the most recent or clean data: we have to search all caches, and hence this should take 135 cycles.

Currently, we are implementing this by making use of sleep(), and giving it an appropriate number. Hence, we have inserted a way to calculate clock ticks. But, we realized we forgot a way to create a use of them. Hence we decided not to work on NUMA first and add storage of issue time of request and the time it is serviced. We had forgot about these numbers initially and realized later that they are important as there is only one bus, and all processors need to share it. And hence, this will lead to contention and increase in the service time, rather than it just making use of sleep() time. This is a little difficult because of the ways memory requests are stored in the simulator, hence we are working on adding it.

Week 3, both of us had on-sites and were gone from Wednesday-Saturday. Because of that, we have been unable to work ahead and add a NUMA implementation. And hence are running behind on schedule.

Running Behind on Schedule
We are running behind on schedule because of our inability to meet up and work together. In all the projects until now we have been doing pair programming, because of us having on-sites, our progress was hampered. And, it will be difficult to catch up this week, because of the 15-418 midterm and Shivani having another on-site.

But this coming weekend our goal is to finish implementing NUMA and make sure the timing design works perfectly on UMA. Once that is done, we will be almost caught up. And since Thanksgiving break is next week, I believe we can get back on track. This is because we did keep a slot of "Finish remaining implementations" on our schedule in week 4. Hence giving us more space.

Poster Session
We will have graphs and demo during the poster session. We will have graphs reporting the data we have collected and the comparison. And, we will have our simulators/emulators up for demo so people can test them themselves and see what the performances are like.

Preliminary Results
We do not have any preliminary results yet.

Big Issues
I believe the biggest issue we will have is combining the implementation of SMP with NUMA architecture to create a hybrid that can take flags from the user and create a custom architecture based on that. We might have to make it less flexible, for the ease of implementation, and that might hinder the number of comparisons we can do the hybrid implementation.

Due to these issues, here is our updated schedule:

Original Schedule
During our project proposal, we had planned a schedule that looked something like this:

Week 1: October 29th - November 4th	• Meet Course Instructors Tuesday(10/29, 10:15 am) • Decide on our project and submit a proposal (10/30, 23:59 hrs) • Study the starter codebase
Week 2: November 5th - November 11th	• Create a way to calculate clock ticks • Added Goal: Create a way to use the the clock ticks by adding issue time and serviced time
Week 3: November 12th- November 18th	**No progress made due to both partners being busy**
PROJECT CHECKPOINT: November 18th	
Week 4: November 19th - November 25th	• Implement NUMA • Generate memory traces
Week 5: November 26th - December 2nd	• Implement UMA+NUMA architecture • Run traces and analyze performance results
Week 6: December 3rd - December 9th	• Work on the final report and project presentation
FINAL PROJECT DUE: December 9th	
PROJECT POSTER SESSION: December 10th	