3 Project Plan

3.1 Project Management/Tracking Procedures

Our project uses a hybrid management style, utilizing both waterfall and agile development strategies. This has allowed us to plan out the entire project broadly, then use agile development strategies to focus on week-to-week goals to meet broad deadlines. To achieve these goals, we are using Git to manage our code and tracking issues using Gitlab Issues. We are also meeting weekly with our client, Boeing, to make sure that we are on the right track and that they are happy with our progress. These strategies allow us to make consistent progress and track where we are relative to our goals.

3.2 Task Decomposition

We divided our project into major parts, then subdivided those parts into smaller parts that can be taken care of by one or two people. This strategy integrates well with our management and tracking procedures. Below is a chart of the tasks that we broke the project into.

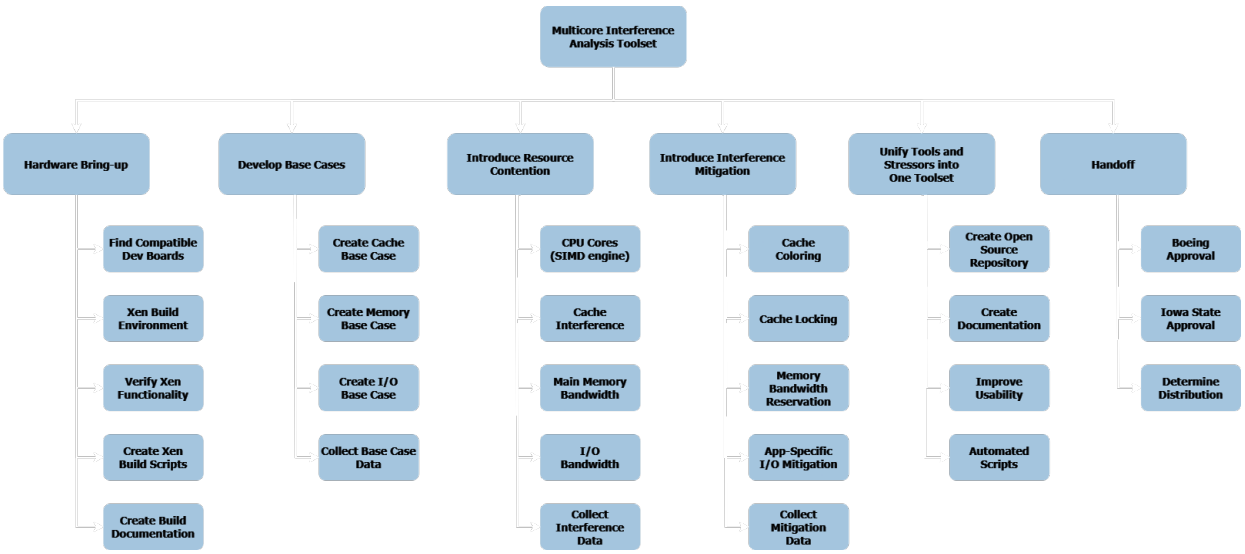

Figure 1: Task Decomposition Chart.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Metrics/Evaluation Criteria:

- Technical metrics:
  - Tool suite generates interference on all aspects of the hardware platform
    - Cache, memory, I/O buses, SIMD engines, etc.
    - Options should be configurable to service multiple different platforms
  - Worst-Case Execution Time (WCET) Criteria
    - Perform several experiments to generate interference
    - Use statistical analysis of execution time to determine an upper bound on WCET
  - System resource usage
    - Characterize the underlying interference channel causing the WCET
  - Same functionality in command-line version of tool suite as the GUI version
- Usability metrics:

- Users rate appearance and usability of the tool suite > 7 on a scale of [1 – 10]
- Command-line version of the tool suite is as user friendly as GUI frontend

Milestones:

- Xen hypervisor is functional on our target development platform
- Identify resource contention points on our target platform
- Tools induce *some* amount of stress on the identified contention points
- Tools thoroughly induce stress on the identified contention points
- Quantitatively prove mitigation tools improve performance of the system while contention is underway
- Obtain a worst-case execution time for our system
- Integration of testing and tools into one unified suite

3.4 PROJECT TIMELINE/SCHEDULE

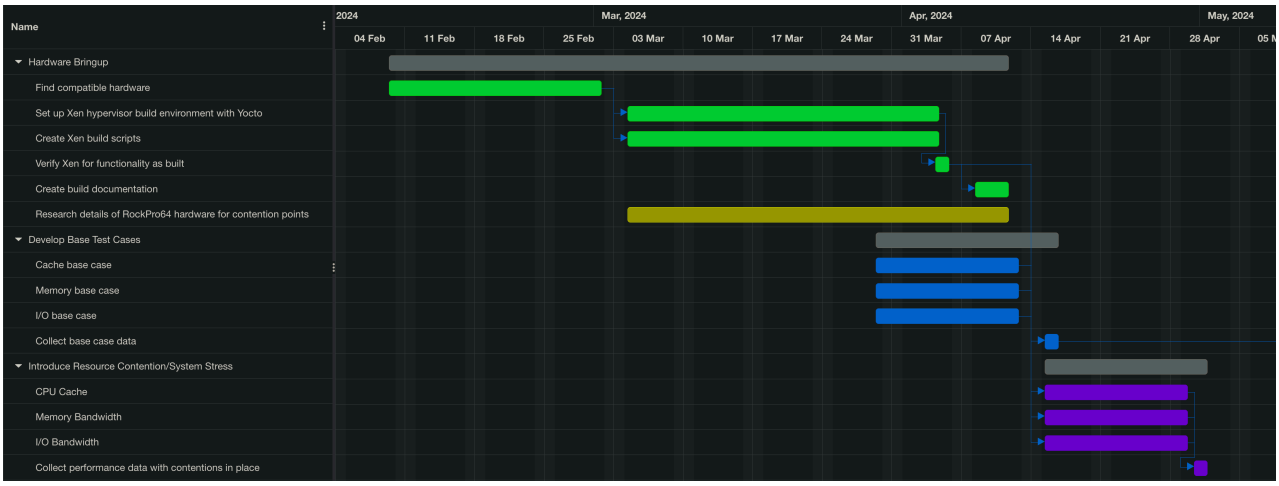

Figure 2: Semester 1 Project Timeline.

For semester 1, we have several deliverables we are planning on targeting. The first is having the hypervisor functional, task "Verify Xen for functionality as built" in the chart and is scheduled for April 5th. The second, identify resource contention points (yellow), comes shortly after as we do research on the board while getting it set up and is targeted for April 11. The third deliverable, inducing stress on the identified contention points, should be finished by April 29th.

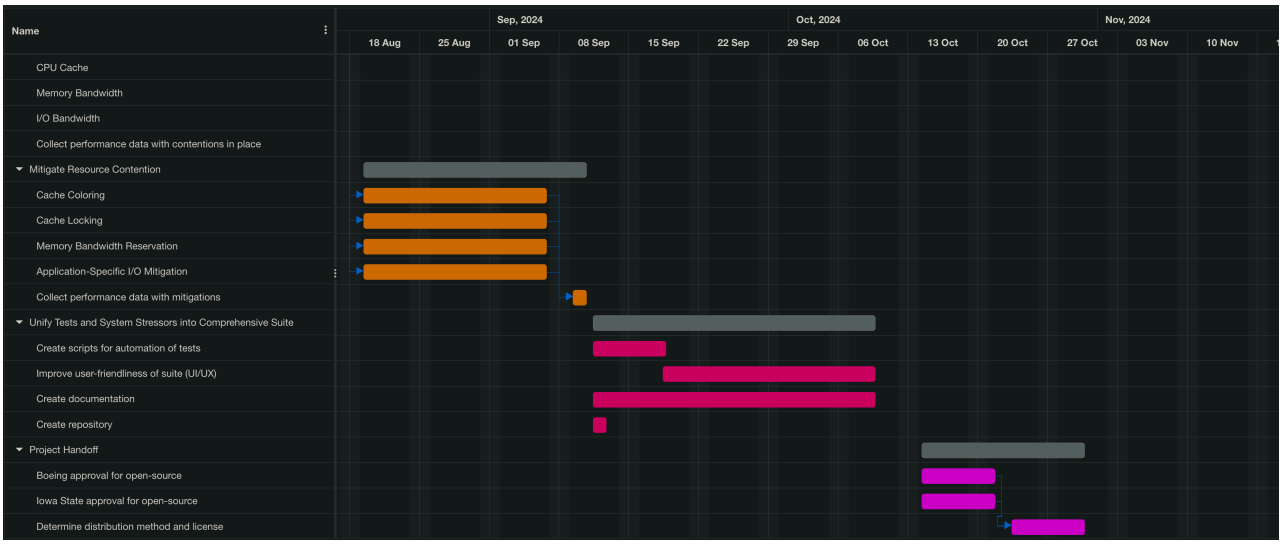

Figure 3: Semester 2 Project Timeline.

For semester two, we have several more deliverables. The first is quantitatively proving that our mitigation techniques improve the execution time of our base case programs, which will happen when we collect the performance data on September 10th. From there, we can determine a bound on worst-case execution time for our system. This will be easiest to do after we have improved the suite's usability (this milestone is targeted for October 9th), so we estimate that it should be completed by October 31st.

3.5 Risks and Risk Management/Mitigation

Hardware selection:

- Find compatible dev board
  - Risk: we struggle to find a board that meets our project's needs
  - Probability: .80
  - Mitigation: acquire multiple boards (within budget) to ideally find one that works
  - Mitigation: we can emulate a hardware environment in Linux
- Xen build environment
  - Risk: we face challenges building Xen & its toolchain for our platform
  - Probability: .40
- Verify Xen functionality
  - Risk: Xen & and its toolchain do not work after installation
  - Probability: > .90
  - Mitigation: communicate our difficulties to our client to get unstuck early when we encounter issues
- Create Xen build scripts
  - Risk: our set up is not easily replicable / portable to a script
  - Probability: < .10
- Risk: the selected hardware platform is incompatible with our client's and project's needs
  - Probability: > .80
- Risk: we encounter trouble installing Xen on our hardware platform
  - Probability: > .80
  - Mitigation: communicate our work and where we are block to the Boeing team to get unstuck

Develop Base Cases:

- Create Cache Base Case:
  - Risk: we cannot find all the relevant information for the cache for our given platform

    - Probability: .20 (we specifically chose platforms for which we would have this information)
- Create Memory Base Case:
    - Risk: we cannot find all the relevant information for the memory configuration for our given platform
    - Probability: .20 (we specifically chose platforms for which we would have this information)
- Create I/O Base Case:
    - Risk: we cannot find all relevant information for the I/O characteristics for our given platform
    - Probability: .20 (we specifically chose platforms for which we would have this information)
- Collect Base Case Data:
    - Risk: we have no way to collect relevant metrics on the researched interference channels
    - Probability: .30

Introduce Resource Contention:

- CPU Cores (SIMD Engine)
    - Risk: properly implementing the interference generator is more time consuming than originally planned
    - Probability: .60
    - Mitigation: communicate our work and where we are block to the Boeing team to get unstuck
- Cache Interference
    - Risk: properly implementing the interference generator is more time consuming than originally planned
    - Probability: .60
    - Mitigation: communicate our work and where we are block to the Boeing team to get unstuck
- Main Memory Bandwidth
    - Risk: properly implementing the interference generator is more time consuming than originally planned
    - Probability: .60
    - Mitigation: communicate our work and where we are block to the Boeing team to get unstuck
- I/O bandwidth
    - Risk: properly implementing the interference generator is more time consuming than originally planned
    - Probability: .60
    - Mitigation: communicate our work and where we are block to the Boeing team to get unstuck
- Collect Interference Data
    - Risk: our test base cases do not adequately stress the system (i.e., demonstrate WCET)
    - Probability: .60
    - Mitigation: we can use our client's expertise in the given domain to increase the likelihood that our test cases demonstrate the WCET for our hardware platform

Introduce Interference Mitigation:

- Cache Coloring
    - Risk: none of the existing cache coloring tools work for our project
    - Probability: .20 (cf, Stress-NG)
- Cache Locking
    - Risk: we have no method of enforcing cache locks in software / hardware
    - Probability: .10
- Memory Bandwidth Reservation
    - Risk: none of the existing memory reservation tools work for our use case
    - Probability: .05 (cf, MemGuard)

- App-Specific I/O Mitigation
    - Risk: the I/O behavior of applications varies widely, and it will be hard to measure
    - Probability: .60
    - Mitigation: Get input from our client on what applications we should use to adequately mitigate this form of interference
- Collect Mitigation Data
    - Risk: we are not able to accurately measure how our mitigation methods reduce interference
    - Probability: .40 (simply compare metrics with interference on / off)

Unify Tools and Stressors into One Toolset:

- Create Open-Source Repository
    - Risk: we are not able to create a public repository due to NDA
    - Probability: > .50 (?)
    - Mitigation: we need to communicate with both Boeing and Iowa State University early on to determine which parts of the project can and cannot be open-sourced
- Create Documentation
    - Risk: development of the project was not continuously documented, and knowledge is lost
    - Probability: .60
    - Mitigation: maintain light documentation of work throughout the project so it can be expanded on during this stage
- Improve Usability
    - Risk: our tool is not intuitive to use for knowledgeable users
    - Probability: .70
    - Mitigation: perform a usability study with our Boeing clients to improve the usability of our project
- Automated Scripts
    - Risk: the scripts we produce are not able to be reused by users of the project
    - Probability: .20

Handoff:

- Boeing Approval
    - Risk: Boeing does not approve the handoff of our project due to NDA
    - Probability: .30
- Iowa State Approval
    - Risk: Iowa State University does not approve the open sourcing of the project
    - Probability: .50
    - Mitigation: Communicate with both Boeing and Iowa State University early on to determine what parts of the project can be open sourced
- Determine Distribution
    - Risk: there is no platform we can publish our project on or no license that applies to its distribution
    - Probability: .10

3.6 Personnel Effort Requirements

Using the task decomposition table from 3.2, we separated the major tasks into the rows of the table below with the smaller sub-categories/tasks placed along the columns. Using our best judgement, we assigned rough time estimates for each sub task using the assumption that a single team member or two would be assigned to each task. As the project is still underway, the times are subject to change.

Hardware Bring-up	Develop Base Cases	Introduce Resource Contention	Introduce Interference Mitigation	Unify Tools and Stressors to One Toolset	Handoff

Compatible Dev Boards (3hrs)	Cache Base Case (5hrs)	CPU Cores (12hrs)	Cache Coloring (10hrs)	Create Open-Source Repository (2hrs)	Boeing Approval (1hr)
Xen Build (35hrs)	Main Memory Base Case (5hrs)	Cache Interference (16hrs)	Cache Locking (16hrs)	Create Documentation (4hrs)	Iowa State Approval (1hr)
Verify Xen (2hrs)	I/O Base Case (5hrs)	Main Memory Bandwidth (12hrs)	Memory bandwidth Reservation (14hrs)	Improve Usability (8hrs)	Determine Distribution (1hr)
Xen Scripts (2hrs)	Collect Base Case Data (8hrs)	I/O Bandwidth (12hrs)	I/O Mitigation (14hrs)	Automated Scripts (8hrs)	
Build Documents (4hrs)		Collect Data (8hrs)	Collect Data (8hrs)		

3.7 Other Resource Requirements

The main requirement for this project is an ARM development board that supports Xen Hypervisor; for our project that is a RockPro64 accompanied with a MicroSD card for booting. In the case of the RockPro64, a USB to TTL converter is also required to communicate with the board over a serial connection. This was primarily purchased to aid with client-to-team debugging. Along with the board, SD card and USB to TTL converter, a computer with a USB type A is also required to interface with the development board.