**Team Introductions**

* Name
* Interest in the NASA JPL project
* Future plans after graduation (?)
* Personal (?), i.e. age, place of birth, pets, etc.
* Team Introductions shouldn’t take longer than 10 seconds each.
  + 10 seconds each, 7 members is about 1min and 10 seconds

**Sponsor Introduction [Section 1]**

* History, background, and mission of NASA JPL  
  + In the beginning it all started as a group of Caltech students who were amateur rocket enthusiasts in the 1930’s. It was not until 1944, having gained sponsorship under the U.S. Army, when the laboratory was officially named the Jet Propulsion Laboratory. Under the Army sponsorship the JPL would develop rocket technology and missile systems of which two specific ones were the Corporal and Sergeant missile systems. A major contribution of JPL was during the space race between the United States and the then Soviet Union. JPL built the Explorer I of which launched on January 1st, 1958 and was the United States’ first satellite and was responsible for detecting the Van Allen radiation belts.
  + On October 1st, 1958, the newly formed National Aeronautics and Space Administration absorbed NACA, the National Advisory Committee for Aeronautics and with it was transferred JPL from the Army.
  + JPL’s focus shifted from rockets themselves to rocket payloads, however despite this change in focus the laboratory’s name was retained. NASA JPL is what is known as a Federally Funded Research and Development Center (FFRDC) and is managed by Caltech but funded through a NASA contract that began in 1958 when the lab was transferred from the Army to NASA.
  + The current mission of NASA JPL is to carry out space and Earth science missions.
* SPICE Library history **[Section 2]**  
  + While no official date is listed for the birth of the SPICE library it’s development started between 1982 and 1983.
  + The SPICE Library was created based on a need brought to light by the National Research Council’s Committee on Data Management and Computation or CODMAC. The report stated the need to be able to properly interpret data from space science instruments with respect to those instruments.
  + The SPICE Library as it’s known today was developed by the Navigation and Ancillary Information Facility (NAIF) under the direction of NASA’s Planetary Science Division for the purpose stated above.
  + The first official use of SPICE was for the Magellan mission to Venus.
* What does the SPICE acronym stand for?  
  *All information is available from “The SPICE Concept” at:*  
   *<*[*https://naif.jpl.nasa.gov/naifspiceconcept.html*](https://naif.jpl.nasa.gov/naifspiceconcept.html)*>*  
  + **S - S**pacecraft ephemeris, given as a function of time.
  + **P - P**lanet, satellite, comet, or asteroid ephemerides, or more generally, location of any target body, given as a function of time.  
      
    The **P** component also logically includes certain physical, dynamical and cartographic constants for target bodies, such as size and shape specifications, and orientation of the spin axis and prime meridian.
  + **I - I**nstrument information containing descriptive data peculiar to the geometric aspects of a particular scientific instrument, such as a field-of-view size, shape, and orientation parameters.
  + **C -** Orientation information containing a transformation, traditionally called the “**C** - matrix,” which provides time-tagged pointing (orientation) angles for a spacecraft bus or a spacecraft structure upon which science instruments are mounted. The C component may also include angular rate data for that structure.
  + **E - E**vents information, summarizing mission activities - both planned and unanticipated. Events data are contained in the SPICE E-kernel file set, which consists of three components: Science Plans, Sequences, and Notes.

**Parallel SPICE Implementation [Section 3]**

* The need for this project  
  + NASA JPL’s SPICE Library is flight-proven and does not require any updates of which is why it is necessary to expand its capabilities while preserving the integrity of the library itself.
  + As it stands, the existing SPICE Observation Geometry System for Space Missions is not sufficient because it is not thread-safe and does not take advantage of multi-core systems to increase performance.
  + The Parallel SPICE Implementation will be a thread-safe wrapper library to increase calculation performance and introduce safe parallel execution while also isolating volatile components.
* Objectives **[Section 4]**  
  + The primary objective of this project is to create a wrapper library for the SPICE Java implementation that will support parallel execution of SPICE functions without modifying the library itself.
  + This wrapper library should provide an increase in performance versus the currently used implementation being used at NASA JPL and therefore must be tested, validated, and benchmarked against the native SPICE.
* Scope  
  + The parallel adapter for the SPICE Java implementation will replicate all of the existing SPICE functions and support parallel task execution by maximizing the use of resources available to the system being used.
  + Result validation testing will be performed to ensure that the wrapper successfully executes tasks in parallel.
  + We will use performance benchmark tests to provide quantitative results to compare with the native SPICE implementation performance.
  + A report of findings and documentation about the Parallel SPICE Implementation and the use of this wrapper library. The main users of the implementation we are constructing are NASA scientists and engineers, not developers so the documentation will provide instructions on how to interact with the wrapper library to get the results they want.
* Project approach   
  + Parallelization research was conducted to search for potential methods of distributing tasks to parallel processors and the products of that research yielded ZeroMQ and gRPC as potential candidates:  
    - ZeroMQ is a high performance asynchronous messaging library and was considered early as a potential way to implement distribution however issues with implementing the library with Java and performance drops were of significant concern.
    - gRPC is Google’s Remote Procedure Call system. Used with Protobuf to distribute messages and data, our team had some familiarity with this system and found that while more complicated than ZeroMQ appeared to have performance benefits that outweighed the complexity introduced.
* gRPC Messaging Architecture **[Section 5]**  
  + Each sub-engine will host its own gRPC server to communicate with the main engine.
  + Google’s Protobuf message template will be used to transport packages between endpoints.
  + SPICE functions will be mapped to a request type and service endpoint to relieve the burden of message routing to already available gRPC tools.
  + To summarize, the main engine will create a request object then dispatch it to a worker where it will be mapped to a service endpoint. That endpoint will invoke the required SPICE function, grab the results, then send that response back to the main engine.
* More on approach (Parallelization)  
  + Construct a small prototype to evaluate the feasibility of the current API candidate with a sampling of SPICE functions.
  + Tests will be completed on the prototype to test the performance and accuracy of the chosen API for performance and accuracy.
  + Once testing on the prototype is complete it will be expanded to include the full SPICE library.
  + Test the parallel SPICE implementation to see how it stacks up against the native SPICE implementation in terms of performance and accuracy.
  + We will need to document how the parallel wrapper library was constructed and how to use the wrapper library for our end-users at NASA JPL.
* Architecture of parSPICE **[Section 6]**  
  + parSPICE uses a master / slave architecture
  + There is a primary engine that manages resource use and task distribution to the workers.
  + Sub-engines or workers will be initiated from the main engine to process tasks and invoke native SPICE methods.
  + parSPICE will provide concurrency similar to thread synchronization but without the complex methods that follow such an implementation. This will eliminate the need to troubleshoot or work around the issues that come with thread synchronization.
  + The native OS’s thread scheduler will be leveraged to handle many of the allocations of the engine and sub-engines to available cores, etc.

Joel’s notes:

Parspice uses a master slave architechture, with a master dispatch engine sending out batch requests to a worker pool. It then collects the results in waves and returns them to the parspice frontend as soon as possible. The consumer will then receive all of the results in one large collection when all of the tasks are complete.

* Deliverables

Joels notes:

For the deliverables, our primary objective is the parallelization, so our first milestone will be prototyping a few spice functions with proper concurrency. We’ve already discussed our choice to use gRPC, and the team is already on the way to a limited prototype. Once we have tested it for correctness, we will expand the api to include all spice functions in the full solution, which will then be tested and benchmarked to ensure we meet the goals set by the sponsor. Lastly in our documentation and report, we will include details on how to navigate and use parSPICE, as well as how the parspice api relates to CSPICE and JNISPICE, and a record of the important design decisions and compromises we made. The report will also contain our performance gain expectations based on the benchmarks, relative to using JNISPICE directly without our library.

* + Parallelization Strategy Research, Prototype, and Testing -   
    - The goal of the early stages of our project were to find candidate API’s for the distribution of messages and our research came down to ZeroMQ and gRPC.
    - After looking through the pros and cons of each candidate we chose gRPC to build a limited prototype for testing against the native SPICE implementation.
    - Testing against NASA JPL’s available solution for performance and accuracy to ensure that our solution is viable and does not take steps backwards.
  + Full Solution Design, Implementation, and Testing -   
    - Once we have a proven prototype we will build upon the initial prototype to design the full implementation of the Parallel SPICE Implementation.
    - This full solution will be expanded to the full SPICE library with distribution, workers, and will take advantage of modern computing resources.
    - Once a full solution is complete it will be tested and benchmarked to ensure we meet the goals set by the sponsor.
  + Documentation and Report -
    - We will need to provide how to navigate and use the Parallel SPICE Implementation and all of its inner workings as well as provide details as to the decisions we made about the construction of the API and why those decisions were made.
    - The report will contain our best, worst, and average case performance expectations based on our benchmarking and make comparisons to the native SPICE implementation that is currently in use.
* Progress Report: Where we are and where we plan to be for the onset of the Spring Semester for our Senior Projects Course. **[Section 7]**  
  + A Project Charter has been completed and agreed upon with our sponsor at NASA JPL to design and construct the Parallel SPICE Implementation.
  + A requirements document has been completed to outline the need and scope for this project. This document also outlines use cases, performance objectives and the functional and non-functional objectives for the library which will be named parSpice.
  + A benchmark performance plan has been put together for the testing of our candidate prototype however is applicable with minor changes to the full parSPICE implementation.
  + We have mapped the scope of the SPICE library that parSPICE is to be implemented for. As of now there is no plan to expand any wrapper functionality to the CSPICE part of the SPICE library.
  + Early testing of the gRPC and ZeroMQ candidates yielded promising results with the selection being gRPC for its ease of use and the performance improvements over ZeroMQ.
  + The design plan for the gRPC dispatch implementation is for it to be scalable to use more or less workers and different batch sizes based on resource availability or limitations.
  + Multiple consumer API’s have been designed and pitched to the sponsor and one was selected based on the needs of NASA JPL.
  + Our team has made strong progress on our parSPICE prototype and are on schedule as per our projections in the Project Charter that was completed.