Project Charter

JPL Parallel SPICE Implementation

CU Boulder 2020-2021 CS Capstone

September 28, 2020

Version 1.0

1. Introduction

Executive Summary

- NASA's JPL requires a parallel wrapper implementation for their existing SPICE Observation Geometry System for Space Science Missions. The current solution operates in a manner that is not threadsafe, and such a solution is required to increase calculation performance.
- The SPICE library is flight-proven, heavily optimized, and algorithmically complex. Thus, the library cannot be modified or reimplemented. This project will produce a threadsafe wrapper library that manages parallel execution while isolating the volatile components.

Business Problems/Opportunities

- The SPICE library was built many years ago and utilizes many global variables in its implementation. Multi-threaded execution was not a concern at the time. This makes it difficult to scale systems with large calculation sets.
- Since modern computers now support many parallel threads, and applications are more often making use of large datasets for 3D visualizations and analytics, a more performant solution is needed. The SPICE library needs to be adapted to better deal with modern software applications.
- Subsequent SPICE versions cannot modify the API of previous versions. The API of parallel SPICE must correspond to the API of the existing JNI SPICE library.

2. Objectives and Scope

Business Objectives

- Create a wrapper around the JPL SPICE Java implementation that supports parallel execution of SPICE functions without modifying the underlying library.
- Test the solution and validate the results against native SPICE[1] operations.
- Produce performance comparison benchmarks against native SPICE operations.
- Produce documentation of the solution.
- Produce a report with project findings and results.

High Level Requirements

- The solution will provide a library to facilitate parallel task execution for existing SPICE functions.
- The solution will provide an intuitive API that allows consumers to easily integrate these parallel features into their applications in a similar manner to which they would integrate native SPICE.
- The solution must support Windows, Linux (at minimum: Redhat, Ubuntu), and MacOS.
- The solution should complete calculation batches significantly faster than the single threaded execution of native SPICE. Similarly, the performance should scale relative to the number of threads being used for execution. A larger number of active cores should reduce execution time[2].
- The solution should produce identical results to the native SPICE implementation for any given function and argument set.

[1] "Native SPICE" in this context refers to the existing Java implementation produced by JPL. It does not refer specifically to the machine-specific native compilations of the SPICE tool that Java references via JNI.

[2] Computers are capable of managing more threads than can be processed at once. As such, the parallel SPICE implementation will scale relative to the number of active cores being utilized, since that is a better measure of the parallel execution capabilities of a device.

Project Scope

In Scope:

- A parallel adapter for the SPICE Java implementation. This wrapper should replicate all existing SPICE functions and support parallel task execution.
- Result validation tests.
- Performance benchmark tests.
- Documentation of the implementation and use of the produced solution
- Report of project findings

Out of Scope:

- Any other non-negotiated features, tools, or documentation.
- Documentation of existing SPICE tools.
- Support for any other SPICE language implementations.

3. Project Approach

General Approach – Solution Delivery Process

- The solution will be constructed in sequential phases
    - Parallelization strategy research. Evaluate potential methods of distributing tasks to parallel processors.
    - Use the best parallel strategy to construct a limited-scope prototype. Use a subset of SPICE functions to evaluate the proposed parallelization implementation and API candidate.
    - Test the initial prototype function set for performance and accuracy.
    - Expand parallelization efforts to all SPICE functions.
    - Test the parallel suite against native SPICE results. Test both performance and accuracy.
    - Document the implementation and usage of the parallel library
    - Create report on project findings
- The first and second phases are most critical to the solution. The prototype must be validated by the team and JPL management before continuation. If any fatal errors are found in the prototype, it will be fixed before the project continues.
- Once the parallelization strategy and prototype implementation are approved, they will remain constant for the duration of the project. Modifying the solution core would require a full project reset and reevaluation, which must be avoided.

Assumptions

- Once a preferred solution methodology is identified, JPL. management will maintain full support for the implementation plan
- Subteams will be formed to distribute solution implementation tasks.

Project Risks and Issues

- If a satisfactory parallelization solution cannot be found during the prototyping phase, the project will stall and potentially even fail. It is critical to the project that a satisfactory task distribution algorithm and consumer API are established early in the project lifecycle.
- There is no risk to the existing SPICE library since it will not be changed by this project. A failure of this project will result in status quo continuity.

Project Changes

- Changes to project scope and requirements will be reviewed by all parties involved. Changes must be approved by a committee of both project team members and the JPL sponsor. The approval committee will evaluate the change and its impacts on scope and timeline.

4. Project Plan

Key Deliverables

1. Project Charter (this document)

2. Parallelization Research

- Research various methods of parallelizing SPICE tasks
- Find satisfactory method of task distribution and message processing

3. Parallelization Strategy Prototype

- Small subset of SPICE features will be implemented to test the parallelization strategy
- Produce consumer API candidate

4. Prototype Test Results

- Prototype will be evaluated for performance and accuracy
- Findings will be furnished to the JPL sponsor for approval

5. Full Solution Design

- A plan will be developed to expand the prototype to cover all SPICE functions
- The design will be furnished to the JPL sponsor for approval

6. Full Solution Implementation

- Project subteams will implement the solution as designed

7. Solution Test Results

- The solution will be evaluated for performance and accuracy
- Findings will be furnished to the JPL sponsor for approval

8. Project Documentation

- Parallel SPICE consumer implementation guide
- Decision document outlining all important project decisions

8. Project Report, containing

- Internal design details such as task distribution, messaging, serialization, etc
- Best, worst, and average case performance expectations
- Comparisons to native SPICE

Timeline

Deliverable	Due Date
Project Charter	October 2, 2020
Requirements Documents	October 9, 2020
Parallelization Research, Strategy Prototype	December 1, 2020
Prototype Validation Testing	December 15, 2020
Full Design Document	January 31, 2020
Full Implementation	March 31, 2020
Solution Validation Testing	April 15, 2020
Project Documentation	April 15, 2020
Project Report	May 1, 2020
Project Management Handover	May 1, 2020

Preliminary Cost Estimates

Labor Costs	**Estimate**
Software development services	Free

Hardware/Software Costs	**Estimate**

There will be no additional software or hardware costs.

5. Key Stakeholder Roles & Responsibilities

Stakeholder	Role/Responsibility
Austin Albert	**Project Manager**
Marcel Llopis	**Executive Sponsor, SME**
Nicholas LaMonica	**Developer / Tester**
Joel Courtney	**Developer / Tester**
Willie Chew	**Developer / Correspondence**
Joseph Ledesma	**Developer / Q&A Manager**
Matthew Cohen	**Developer / Tester**
Sahib Bajwa	**Developer / Tester**