Work Breakdown

JPL Parallel SPICE Implementation

CU Boulder 2020-2021 CS Capstone

December 03, 2020

Version 1.0

- Initial simple prototype (fast, easy, c#) - Austin Albert
    - Decide network communication framework
    - Make networking proof of concept
    - Make task distribution POC
- Port simple prototype to java - Austin, Nick
    - Copy behavior from original prototype, but using jdk 8
- Add SPICE behavior to Java prototype
    - SPICE call marshalling over gRPC - Willie Chew
        - Designed preliminary gRPC end points for the client handler and the worker service
        - Designed classes to enable the “bundling” of SPICE calls for concurrent processing
        - Created a proto file describing a GRPC service for different SPICE function calls.
    - Extensible task distribution in Java
        - Support grpc networking calls
        - Support SPICE marshalling subsystem
- Benchmark prototype - Joel
    - Design basic structure (done)
    - Implement specific benchmark tasks (1 complete)
    - Choose most reliable task for final version
- System Review - Austin
    - Analyze SPICE marshalling system
    - Analyze task distribution platform
    - Plan for prototype integration
- Prototype integration - Austin
- Testing Phase 1 - Austin, Joe
    - Test results from original Spice compared to parSpice
    - Test benchmark suit; ensure parSpice performs better than Spice in a multithreaded environment
    - Write unit tests around initial system to prevent degradation
- Automation - Joel
    - Prepare list of CSPICE functions (including name, return type, arguments, pointer-arguments, and distribution type)
    - Create modular approach to generation using Marshalling work
    - Integrate with build chain
    - Automate unit test generation for each function as well
- Testing phase 2 - Joe
    - Create unit tests for all SPICE functions
    - Validate test results

    - Validate benchmark suite
- Project finalization and handoff - Team