SJSU, CmpE 275, Spring 2022, Project Charter

Goals

Code named **Space Opera**, the charter of the project is to explore concepts of distributed, volatile networks of compute processes and how to manage data movement to maximize stability in a non-centralized coordination.

Description

As a member of **Space Opera (SP)** you are tasked with creating a design and with providing a prototype implementation of a scalable network of peer-based processes for cataloging data for high volume retrieval and fault recovery that could include loss of nodes, version resolution, parallel access, and recovery/replication. The network should allow for dynamic growth and membership to support failure and capacity scaling. Likely scenarios would include process (service) deployment management as well as service evolution (change management).

As project members, you must balance everyone's technical endeavors to provide a space to explore interests whilst producing a functioning design.

Challenge areas: Peer-based delegation, self-organizing, peer validation, performance, deployment scaffolding, and scalable adaptive traffic management. This is quite vague in definition and direction; part of your work is to create requirements and a demonstratable solution that meets the target goals.

Near-term goals should be identified and tracked to ensure SP can move forward rapidly. Choosing a methodology for rapid prototyping maybe required to achieve your goals in the short time remaining. For instance, using multiple sub-teams to parallelize coding technical solutions with mini-cyclic design-code-integrate development. I cannot emphasize enough, creating a master design before coding will not work with this project.

Technical Provenance and Scope

The tools and code used to develop the project should include original work as well as available (emerging and established) sources. As stated, the goal of SP is in part understanding how things work and how to stress and identify for improvement perceived weaknesses, platform restrictions of existing solutions. Thus, implementation should span multiple (at least 3) languages. Choosing a tool and language should include criteria for industry applicability, emerging-ness, and performance factors (scaling, concurrency, speed, etc).

Technical scope should include an ecosystem of multiple vendors of distributed and scalable storage, ability to handle high volume traffic/messaging, a deployment solution for services, multi-language support, and non-centralized provenance/agreement.

While Python will weigh heavily for many people in choosing a language, you should understand that it is potentially in direct conflict with SP's goal of performance (ref. Python's GIL).

Data

Much of the core system design can be independent of the data. Factors of data size, frequency of inbound and outbound changes/reads/deletes are more important. Therefore, choosing a data set to validate SP is important for repeatable verification and testing; actual content is less important.

Literature References

Looking for support for your design and implementation may require searching for papers and discussions regarding the technologies and algorithms use in the design. In many cases there is likely multiple sources that discuss case studies and approaches like what you are trying to achieve. Document these. There is also the possibility that you may have to extrapolate a papers' ideas to your unique technical challenge/situation. Document these as well and discuss in your reports how this was achieved.

Testing and Verification/Validation

It is important to choose data that can repeatably create a measured and verifiable result to ensure your algorithm and code functions correctly. This goes beyond simple unit testing. Test cases and data are at the heart of confidence acceptance.

Test your design for multiple cases that you may encounter in an actual deployment, this may include invalid or missing data, invalid authority, time to complete, varying volume (kbytes), varying rates (requests/sec), recovery verification.