Notes on SpaceOS & Path-trailing

Taha Rhaouti

April, 2024

Objectives

These notes are intended to tackle the safety, design, and implementation aspects of SpaceOS, a spatially-aware operating system. We will explore its objectives, research areas, potential applications, and inspiration from the Spatial Web.

Overview

SpaceOS is a proof of concept for a new operating system that leverages spatial configurations to optimize computing performance. This project explores the application of transportation theory, facility location problems, and other advanced mathematical techniques the path-trailing part of the system. The goal is to learn/implement caching, and resource management in distributed systems and networks.

Objectives

- Investigate optimizations in compute speed and caching by utilizing the spatial attributes of the operating system.
- Explore solutions ranging from simple discrete problems solvable using Linear Programming to more complex problems involving differential geometry and probabilistic methods.

Research Areas

- **Transportation Theory**: Applying principles of transportation theory to optimize data and resource movement within the system.
- **Facility Location Problems**: Investigating how facility location algorithms can improve system efficiency and reduce redundancy.
- **Differential Geometry**: Utilizing concepts from differential geometry to handle complex spatial configurations and resource-tolling calculations.
- **Probabilistic Methods**: Implementing probabilistic "games" to simplify and enhance resource management and computational efficiency.

Potential Applications

- **Compute Speed Optimization**: Reducing computation times through efficient spatial data management.
- **Caching and Redundancy Minimization**: Enhancing data caching mechanisms to minimize redundancy and improve access times.
- **Resource Management**: Optimizing resource allocation and usage in distributed systems and networks.

Inspiration from Spatial Web

A stateful Spatial Web enables smart digital twins of people, physical spaces, and objects to be reliably and securely linked together, spatially. The effect of this is that when an object or person moves into or out of any physical or virtual space, a Spatial Contract can be executed automatically, subject to a set of spatial permissions set by the owner or approved entity triggering a record of the action and/or initiating a transaction. This makes the Spatial Web a trustworthy network for any form of interaction, transaction, or transportation.

Smart Spaces

A Smart Space is a defined location— a virtual or physical "place" described by its boundaries, some descriptive and classification information, a Spatial Domain, and a set of interaction and transactional rules (Smart Contracts). Smart Spaces are "programmable space." They are semantically aware and can reference and validate the permissions related to users or assets within them. They can be securely encrypted by Distributed Ledgers and can control what users, objects, software, or robotics are able to be used.

Problem and Solution

Problem: Currently, there is no way to reliably assign Spatial Rights or Permissions management for Users, AI, Spatial Content, or IoT devices because there is no standard method to identify, locate, and assign permissions for activities.
Solution: Enable any space to become a Smart Space whose boundaries are defined by coordinates—either real-world (latitude, longitude, and elevation/altitude) with 0,0,0 or virtual (x/y/z) including outdoor and indoor spaces. Enable for sub-millimeter granularity and third-party re-localization optimization. Smart Spaces enable assets to have proof of their location, ownership, and permissions in time and space, across any device, platform, and location within virtual spaces and in the real world. They are searchable and can transact with Users or Assets. They can support multiple Users and Channels of Spatial Content.
Benefit: This solution enables multiple users to search, track, interact, and collaborate with Smart Assets across time and space within Smart Spaces (i.e., in virtual and geo-locations). Smart Spaces are programmable.

Example

A couple interested in buying a home in another state virtually walks through the various rooms of potential houses and can place their furniture in it to see how it fits. The Port of Long Beach notifies a buyer's account that the cargo that left Hong Kong has just arrived. The buyer's account automatically pays the shipper minus the port fees.

SpaceOS Structure

Space Creation

```
x = create space attributes
y = create space attributes
attributes = { loc:  [x, y], size:  [w, h], type:
_, ...}
```

Path Creation

```
p = create path attribs_p
attribs_p = {start:  x, end:  y}
```

Path-Trail Mapping

To construct the path, we introduce WAYPOINTS, which can be introduced in two different ways: (a) an array of space instances, (b) specifying a space factory that produces the space instances.

Space locations have definite locations and sizes. Space factories can be parameterized to construct space instances that have specific characteristics.

Example

```
p = create path attribs_p = {start:  x, end:  y,
waypoints:  [a, b, c], ..}
a = create space attribs
b=..., c=...
```

Factory z is generating the waypoints automatically.

```
q = create path attribs_p = {start:  x, end:  y,
waypoints:  z, ...}
```

Constraints for Space Factory

```
z = create spacefactory attribs_f
attribs_f = {xline:  [N, M], yline:  [N, M],
type:  _, xstart:  _, xstep:  _, ystart:  _,
ystep:  _, ...}
```

The space factory constrains the waypoints. It can also create multiple paths, connecting the spaces for load balance, minimum hazard paths, etc.

Space Classification

A space is directly created using a space factory. We can specify a main function to execute in a space. This is the controller. Each space runs a single controller while it can have many services. Spaces can be imaginary (i-space) or real (r-space).

Imaginary Spaces (i-space)

These are spaces that do not have a concrete physical realization. A portion of a physical space can be set as an i-space for a certain task.

Real Spaces (r-space)

These associate with the real physical world and can be closed or open (permission management, encryption). For example, a parking space or storage locker.

Chain-of-Thought Prompting

Chain-of-thought prompting has several attractive properties as an approach for facilitating reasoning in language models.

- First, chain of thought, in principle, allows models to decompose multi-step problems into intermediate steps, which means that additional computation can be allocated to problems that require more reasoning steps.
- Second, a chain of thought provides an interpretable window into the behavior of the model, suggesting how it might have arrived at a particular answer and providing opportunities to debug where the reasoning path went wrong (although fully characterizing a model's computations that support an answer remains an open question).
- Third, chain-of-thought reasoning can be used for tasks such as math word problems, commonsense reasoning, and symbolic manipulation, and is potentially applicable (at least in principle) to any task that humans can solve via language.
- Finally, chain-of-thought reasoning can be readily elicited in sufficiently large off-the-shelf language models simply by including examples of chain of thought sequences into the exemplars of few-shot prompting

Another potential benefit of chain-of-thought prompting could simply be that such prompts allow the model to better access relevant knowledge acquired during pretraining. Therefore, we test an alternative configura- tion where the chain of thought prompt is only given after the answer, isolating whether the model actually depends on the produced chain of thought to give the final answer. This variant performs about the same as the baseline, which suggests that the sequential reasoning embodied in the chain of thought is useful for reasons beyond just activating knowledge.

Symbolic Reasoning

Chain-of-thought prompting not only enables language models to perform symbolic reasoning tasks that are challenging in the standard prompting setting, but also facilitates length generalization to inference-time inputs longer than those seen in the few-shot exemplars. Examples:

• Last letter concatenation. This task asks the model to concatenate the last letters of words in a name (e.g., "Amy Brown" → "yn"). It is a more challenging version of first letter concatenation, which language models can already perform without chain of thought.3 We generate full names by randomly concatenating names from the top one-thousand first and last names from name census data (https://namecensus.com/). • Coin flip. This task asks the model to answer whether a coin is still heads up after people either flip or don't flip the coin (e.g., "A coin is heads up. Phoebe flips the coin. Osvaldo does not flip the coin. Is the coin still heads up?" → "no").

Takeaways

Question: I am having a hard time relating Chain-of-Thought and Symbolic reasoning with the problem I want to solve, i.e SpaceOS and the path-trailing system, for finding the best paths between objects. (is that even the problem? idk anymore) maybe instead, we can try and look for a neat application of chain-of-thought into the space system.

Agentic AI

I researchers and companies have recently begun to develop increasingly agentic AI systems: systems that adaptably pursue complex goals using reasoning and with limited direct supervision. 1 For example, a user could ask an agentic personal assistant to "help me bake a good chocolate cake tonight," and the system would respond by figuring out the ingredients needed, finding vendors to buy ingredients, and having the ingredients delivered to their doorstep along with a printed recipe. Agentic AI systems are distinct from more limited AI systems (like image generation or question-answering language models) because they are capable of a wide range of actions and are reliable enough that, in certain defined circumstances, a reasonable user could trust them to effectively and autonomously act on complex goals on their behalf. This trend towards agency may both substantially expand the helpful uses of AI systems, and introduce a range of new technical and social challenges.

Agentic AI systems could dramatically increase users' abilities to get more done in their lives with less effort. This could involve completing tasks beyond the users' skill sets, like specialized coding. Agentic systems could also benefit users by enabling them to partially or fully offload tasks that they already know how to do, meaning the tasks can get done more cheaply, quickly, and at greater scale. So long as these benefits exceed the cost of setting up and safely operating an agentic system, agentic systems can be a substantial boon for individuals and society

Important: Balance between safety/risk and cost of actions.