What Hast Earth to do With Mars: An explanation of Urbit's IPC interfaces

~mopfel-winrux
Native Planet

Abstract

In this groundbreaking paper, we embark on a cosmic journey with the renowned character Ernest P. Worrell as he ventures into the unexplored realm of Martian computing. Drawing inspiration from Ernest's comically ingenious encounters with everyday challenges, we investigate the foundations of what we term "Artificial Stupidity." As Ernest grapples with Martian technology, we delve into the intricacies of programming errors, algorithmic missteps, and the curious phenomena that arise when human-like intelligence meets extraterrestrial computing systems. Our analysis sheds light on the unexpected intersections between humor, artificial intelligence, and the cosmic absurdity of Martian software. Join us in this interplanetary exploration as we unravel the mysteries of Artificial Stupidity through the lens of Ernest's interstellar escapades.

Contents

Manuscript submitted for review.

Address author correspondence to ~mopfel-winrux.

1 Introduction

In the realm of software, two worlds exist in stark contrast: Earth, the familiar domain of established software paradigms, bustling with complex and varied digital ecosystems, and Mars, the enigmatic realm of Urbit, a system as pristine and archaic as the Martian landscape itself. This article sets forth on an expedition to unravel the mysteries of this Martian software landscape, exploring how the intricate processes of terrestrial technology can establish a dialogue with the Martian code.

Unlike Earth, where software is a tapestry of evolution and patchwork, Mars presents a realm of purity. It is a world where software exists as a Maxwellian construct, untouched and unaltered by the non-Maxwellian intricacies that characterize Earth's digital environments. Yet, in this isolation lies a profound awareness - an understanding by the Martians that there exist diverse forms of computing with which they must engage. Mars, in its wisdom, accommodates these external entities, allowing them to interact in a dialect familiar to its inhabitants: the language of nouns.

Our journey into this uncharted territory begins with a deep dive into the architecture of Urbit, focusing on the foundational layer upon which Arvo, Urbit's kernel, operates. This exploration will illuminate the operational dynamics and philosophical underpinnings that make Urbit a universe apart. Following this, we delve into the intricacies of two pivotal components of Urbit's ecosystem: `%khan` and `%lick`. These vanes serve as distinct conduits between the terrestrial and the Martian, facilitating not just interaction but a nuanced form of control and collaboration with Arvo. Through this exploration, we aim to demystify the enigma of Urbit and bridge the cosmic gap between Earth's established software paradigms and the Martian vision of computational purity and simplicity.

2 Urbit's Unique Landscape

2.1 Arvo: The Heart of Martian Computing

Arvo, also known as Urbit OS, represents a paradigm shift in operating system design. Unlike traditional operating systems that operate on preemptive multitasking and complex event networks, Arvo is a purely functional operating system, characterized by its deterministic nature and compact size. The entire Urbit stack is about 30,000 lines of code, with Arvo itself being only around 1,000 lines. This small codebase is intentional, reflecting a philosophy that system administration complexity is directly proportional to code size.

Arvo's design philosophy positions it not just as an operating system but as a new frontier in the peer-to-peer internet space. Unlike traditional operating systems like Windows, macOS, or Linux, Arvo doesn't aim to replace them; instead, it offers a unique user experience akin to a web browser, yet it is a fully-fledged OS in its own right. This distinctive approach enables Arvo to function within a virtual machine, theoretically capable of running on bare metal.

Arvo's unique architecture avoids what is referred to as "event spaghetti" by maintaining a clear causal chain for every computation. Each chain begins with a Unix I/O

event and progresses through a series of steps until the computation's terminal cause. This deterministic nature allows for a high level of predictability and control, a stark contrast to the non-deterministic nature of most Earth-based operating systems.

The kernel's design as a "purely functional operating system" – or more accurately, an "operating function" – underscores its uniqueness. Arvo operates on the principle that the current state is a pure function of its event log, a record of every action ever performed. This determinism is a significant deviation from the norm, where operating systems allow for programmatic alteration of global variables affecting other programs.

Arvo handles nondeterminism in an innovative way. The system, in essence, behaves like a stateful packet transceiver – events are processed, but there's no guarantee of completion, mirroring the behavior of packet dropping in networking. This approach to handling nondeterminism is unique and aligns with Arvo's overall philosophy of simplicity and determinism.

Arvo's determinism is a key feature, stacking it atop a frozen instruction set known as Nock. This concept, though new to operating systems, is not foreign to computing. Similar to how CPU instruction sets like x86-64 are frozen at the chip level, Arvo freezes the instruction set at a higher level, enabling deterministic computation. This high-level determinism contrasts sharply with the non-determinism typically found in Earth's operating systems.

Handling nondeterminism in Arvo is akin to a heuristic decision-making process similar to dropping a packet in networking. This approach views Arvo as a stateful packet transceiver, where the completion of events is not guaranteed, a stark difference from traditional computing models. Additionally, because Arvo runs on a VM, it can obtain nondeterministic information, such as stack traces from infinite loops, through the interpreter beneath it, further enhancing its operational capabilities.

The vision of Urbit, with Arvo at its core, is to transition from developer-hosted web services on multiple foreign servers to self-hosted applications on a personal server. Each architectural decision in Arvo aligns with this vision, emphasizing user ownership and management of data. This approach sets it apart from the conventional model of cloud-based services and centralized data management prevalent in Earth-based systems.

2.2 Vere: The Substrate of Martian Technology

Vere, the runtime of Urbit, plays a critical role in actualizing the Martian computing model on Earth-based hardware. As the Nock interpreter written in C, Vere is intricately optimized to run Arvo, ensuring seamless translation of its deterministic operations into practical execution on conventional systems.

Key to Vere's functionality is how it processes events. Events from the Unix environment are passed to Vere, which then translates and injects them into Arvo. This mechanism demonstrates a profound integration between Vere and Arvo, where Vere possesses direct knowledge of Arvo's state. Notably, Vere can 'scry' or query Arvo's state without altering it, retrieving information as needed. This capability allows Vere to maintain the integrity of Arvo's deterministic model while enabling dynamic in-

teraction with the outside world.

Urbit's use of Unix as its BIOS is a pivotal aspect of its runtime operation. By leveraging the robust, widely-used Unix system, Vere ensures that Urbit can run on a broad range of hardware platforms, effectively making Urbit's Martian technology accessible and operable in Earth's diverse computing environments. This strategic use of Unix not only provides the necessary I/O and optimizations for Arvo but also ensures that Urbit's advanced and unique software architecture can function in tandem with the existing, established hardware and software ecosystems of Earth.

Moreover, Vere's design allows it to inject events into Arvo, a feature crucial for maintaining the fluid communication and interaction between the Urbit system and its underlying hardware and software infrastructure. This bidirectional communication channel ensures that Arvo can respond to external stimuli while remaining true to its functional and deterministic nature.

Thus, Vere serves as more than just a bridge between Martian technology and Earthly hardware; it is a finely tuned conduit, optimized to uphold and facilitate the unique computational paradigm that Arvo embodies.

3 Background and Literature

Exposite the relevant background and literature.

1. Prefer to `enumerate`.
2. It is easier to refer to than `itemize`.

Mark `\noindent` in paragraphs that continue the thought of an `\enumerate` block.

4 Urbit's Implementation

Oftentimes you will then turn to exploring how Urbit or closely related systems like Nockchain have approached or considered a problem.

```
#include <stdio.h>

double compute(float a, float b) {
    return a * a + b * b;
}

int main() {
    float x = 2.5;
    float y = 3.7;

    double result = compute(x, y);

    printf("Result: %lf\n", result);

    return 0;
}
```

Listing 1: Example Python Code

```
def hello_world():
    print("Hello, world!")
```

That code snippet was part of the text, and not a standalone entity to which we make separate repeated reference.

Refer to the code listing: Listing 1. (Note that we don't suppress indentation since it's a float.)

5 A Wild Ernest Appears

You can have as many sections as make sense. Only sections and subsections appear in the table of contents.

6 Conclusion

To summarize, why did you write it? Why do we care? What impact should it have on Urbit development?

Your bibliography is a separate BibTeX file. We use the `plainnat` bibliography style. You can use `natbib` citation commands like `\citep{wikipedia}` for parenthesized references. Use `\citet{wikipedia}` for inline references. You can also use `\citeauthor{wikipedia}` for the principal author's name.

"You can use traditional TeX—or LaTeX—representations" [Varney, 1987].

"Or you can use fancy quotes—and symbols."

References

Jim Varney. Ernest goes to camp. Film, 1987. URL `https://www.imdb.com/title/tt0092974/`. Directed by John R. Cherry III. Produced by Stacy Williams. Written by John R. Cherry III and Coke Sams. Starring Jim Varney, Victoria Racimo, John Vernon, Iron Eyes Cody, Lyle Alzado, Gailard Sartain, Daniel Butler, Patrick Day, Scott Menville, Jacob Vargas, and Todd Loyd. Buena Vista Pictures.