Editor: Brian Brannon ■ bbrannon@computer.org

Scaling Virtual Worlds with a Physical Metaphor

Daniel Horn, Ewen Cheslack-Postava, Tahir Azim, Michael J. Freedman, and Philip Levis

The Web has revolutionized computing applications. Originally designed to navigate static documents using simple browsers and servers, the Web evolved into a full-fledged application platform backed by a myriad of technologies. This transformation also fundamentally changed the underlying network. Without Web applications, we might not have layer-7 switches, SSL, or content distribution networks. The Web's need for complex, distributed services similarly changed computer systems, leading to the "computing clouds" of large data centers.

The Web's gradual evolution has led to complex combinations of many underlying technologies. While the ability to flexibly stitch together databases, Ajax, and other technologies has been empowering, constituent parts can interact in unforeseen and dangerous ways. SQL injection is a canonical example: a Web service that does not carefully filter user input is vulnerable to users writing arbitrary queries against back-end databases. A clean and thought-out design would undoubtedly have been simpler and safer.

We believe that one of the next major application platforms will be three-dimensional, online virtual worlds. They provide a compelling substrate for shared, networked environments where people can communicate, shop, socialize, collaborate, and learn. Applications of virtual worlds are already gaining traction. Numerous multiplayer online games such as World of Warcraft, Everquest, Lineage, and Eve Online demonstrate that virtual worlds are a lucrative and powerful platform for entertainment. An ever-growing list of blue-chip companies are deploying their own worlds, as evidenced by Intel's and IBM's research on virtual worlds, Sony's Home, and Sun's Project Wonderland.

Unfortunately, the early evolution of virtual worlds is as ad hoc as the evolution of the Web. Systems have completely independent constructions, sharing few architectural aspects and offering no interoperability. Users generate content with custom formats, new world-specific programming languages are created for programmable behaviors, and proprietary protocols run each world. Systems today are closed, limited, or do not scale. The problem of designing open, programmable, scalable, secure, and extensible virtual worlds remains an open research problem.

The problem of designing open, programmable, scalable, secure, and extensible virtual worlds remains an open research problem.

The Meru Project at Stanford University is designing and implementing an architecture for the virtual worlds of the future. The hope is that we can avoid some of the complexities the Web has encountered by learning how to build applications and services before they are subject to the short-term necessities of commercial development. While Meru cannot compete with the content creation of commercial virtual worlds, it can, like the original World Wide Web at CERN, investigate basic questions about system design. By doing so, we can open the door to a future where physical sensors in the real world seed their virtual reflections, users can visually browse a sea of information, and virtual avatars convey physical social cues to bring distance interaction to the level of actual presence.

Our research suggests that we can achieve these properties by a careful architecture of the underlying systems and using the physical world as analogy for communication within the virtual world. By creating a special *space zero* that reflects the physical world and by constraining digital communication based on real-world physics, we can bridge virtual and physical environments at a planetary scale.

COMPONENTIZING A VIRTUAL WORLD SYSTEM

Virtual worlds today spend significant effort to scale to large numbers of users. Approaches range from *sharding* (where different groups of users inhabit different replicas of the entire world),

Published by the IEEE CS ■ 1536-1268/09/$25.00 © 2009 IEEE

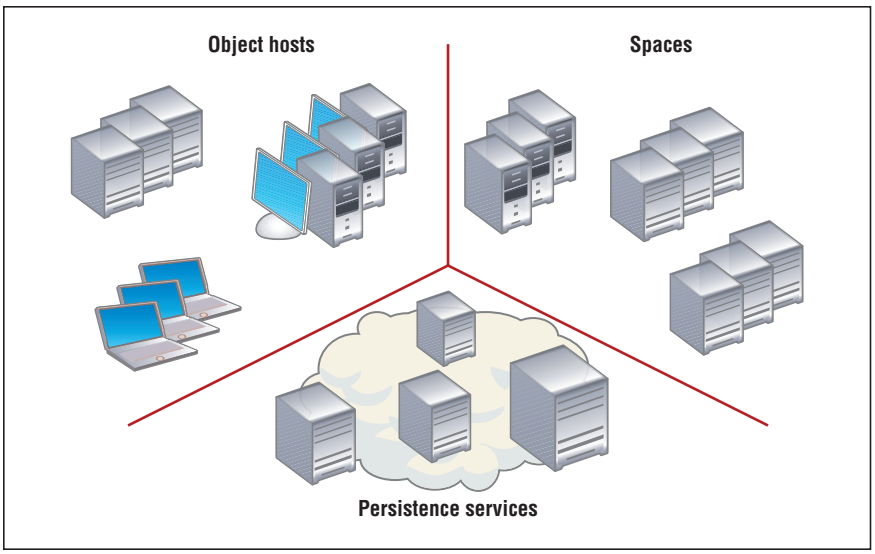

Figure 1. The components of the Meru architecture. Object hosts run code for active behaviors, persistence services store large, static data, and spaces manage the 3D address space (geometry) of a world.

to *instancing* (where a given world has multiple copies of a single place), to custom hardware. For example, World of Warcraft uses a combination of sharding and instancing. Eve Online, in contrast, has a single, always shared world. Eve Online achieves this by building custom multimillion-dollar server racks with battery-backed RAM. To provide persistence, Eve has one hour of downtime each day for flushing RAM to a non-volatile backing store.

Because these worlds are all walled gardens based on paying customers, they can architect their systems around the desired quality-of-service of their real-time user experience. In the future, however, when anyone and everyone might have their own virtual world, we need more flexible and open mechanisms. While some worlds might need to scale to millions of users, there will be small, private worlds as well. Small virtual offices for two- or three-person meetings should be able to run as a process on a desktop, while virtual stores could have backing corporate IT infrastructure.

Second Life presents an interesting contrast to this approach. The Second Life world is statically divided into regions of land called *sims*. Each sim is statically assigned to a server (one server can host multiple sims). This static allocation provides a strong minimum quality of service to a given sim, but as popularity follows a Zipf curve,[1] it results in most computational resources being idle.

Architecture

The Meru architecture focuses on three important properties: expansibility, federation, and migration. Expansibility means that worlds are not architecturally limited in size, and that expanding and adding worlds is easy. Supporting federation recognizes that like the Web, the universe of worlds will be run by a multitude of administrative entities on different computing infrastructures. Finally, because load in these worlds can vary greatly both in time and in space, the ability to efficiently and quickly migrate load or tasks across the underlying systems is critical.

Figure 1 shows how the Meru architecture takes the tasks of a virtual world server and breaks them into three distinct services: computation (object hosts), communication (spaces), and storage (persistence services). The core architectural split is between objects, which perform the computation to simulate individual object behavior, and spaces, which are the 3D medium of communication. A space is a 3D world, hosted by one or more space servers. Long-term, highly shared data, such as meshes and textures, are stored in a separate persistent storage system.

Object hosts simulate objects. Objects can move across hosts, and a given object host can handle objects from many different spaces. Similarly, many different object hosts can handle the objects of a single space. On one hand, locality means that nearby objects should be on the same or nearby object hosts. On the other hand, users might wish to control where their code executes. This tension between using virtual and physical resources reflects the common trade-off between performance and security. The Meru architecture does not take a position on this trade-off, leaving it to individual object and object host policies to decide. A common scenario could be for an enterprise user to host their objects on their company's network, keeping potentially sensitive data secure, while connecting to an external space and maintaining their corporate identity. This separation allows a user to keep their data and objects private while still participating in a public world.

While object hosts store runtime data, such as object variables, *persistence services* store the large, read-mostly data a virtual world needs, such as textures and meshes. Meru splits persistence services into a number of subcomponents. Objects with read-mostly workloads, such as bitmaps and meshes, are stored in a content distribution network (CDN), which is optimized for availability, longevity, and throughput. This offloads most storage from space servers and object hosts, which no longer require a large storage component. The CDN may be scaled using known techniques.[2] For other kinds of persistent data that cannot be effectively stored in a CDN, such as financial data with strict ACID requirements or objects' location information with small and frequent updates, Meru

SPOTLIGHT

allows the object hosts to use their own persistence solutions.

Space: Communication

Spaces are the key coordinator of the Meru architecture. A space represents a virtual world and provides all of the associated services. First and foremost among them is communication between objects: all messages between objects pass through a space. The space authorizes objects for entry and assigns unique references for communication. In order for objects to communicate, a space's spatial query service must introduce them to each other. This dependence on proximity and introduction is in contrast to the "default on" model of traditional Internet communication and can protect against spam and denial-of-service attacks.

The space is responsible for more than just communication. It also enforces the rules of the physical space and ensures a consistent view of the world is presented to all objects. The particular rules enforced depend on the type of world, but a common case is simple collision detection and resolution. Providing this service implies that the space is authoritative on some critical, space-specific state for each object such as location and mesh collisions.

Finally, a space server can include services that it can provide more efficiently. For instance, by leveraging the location information it already maintains, the space can efficiently mix and deliver per-object audio streams, reducing the overall cost of an audio chat service. One current area of our research is to understand which services need sufficiently fine-grained location information, such that incorporating them into a space server is necessary.

Because spaces clearly play a central role in the consistency and performance of a virtual world, a scalable design is crucial. An enormous world is a single space spread across many separate servers, each of which needs to communicate with others to pass messages between objects. Later in this article, we discuss how Meru uses the cues and physical limitations of real-world physics to enable space communication to scale to enormous worlds. First, however, we touch on a special space in the Meru architecture: the real world.

INTERACTING WITH THE REAL WORLD: SPACE ZERO

Given this architecture for a virtual world system, how can we allow real, physical objects to access and interact with the virtual world? And conversely, how can we allow virtual objects and remote users to query and interact with objects in the real world? Not only does such functionality provide compelling use cases for Meru, it also introduces a scalability challenge if the real-world's virtual mirror is similarly large and seamless.

Meru reserves a special space with identifier 0, called space zero. Real-world objects can register with space zero, thus advertising their presence and location in the real world to users of the virtual world. Placing virtual representations of physical objects in space zero provides an elegant way for users to query and interact with the physical world. The same fundamental services provided by any virtual space are also provided by space zero: authorization, introduction, communication, and spatial querying.

As an example of how this mirror world could be used, consider a user of space zero walking down a (real-world) city street. That user's smartphone might register an avatar in space zero, continuously updating its position via GPS. Shop owners might register objects for their stores, also providing their location. Because the world is real-time, shop owners can advertise current specials with virtual sale signs. The user can then query for nearby stores and retrieve a set of descriptions, filtering for particular types, such as restaurants. Businesses could even use per-object sales and custom pricing to their advantage to gauge better pricing points as they register queries from the avatars, which could translate into physical visits. One important point here is that, just as with the Web, while virtual worlds could be designed for human viewing, bots and other automated objects will comb through their data to index and search.

This reflection of the entire physical world into a single space has several important implications. First, unlike most other spaces, space zero is governed by multiple administrative entities—the nations and states that constitute physical space—although space zero might use an overriding logical administrative entity to assist with federation (analogous to ICANN for virtual worlds).

Second, the services of the space must seamlessly scale to planetary levels, including location, query, and communication services. The sheer flexibility afforded by virtual world systems is both enabling and challenging: it allows objects to break the limits of physical reality, but also makes it nontrivial to build a system which is simultaneously scalable, robust, and reliable. Indeed, even a smaller-scale virtual space controlled by a single administrative entity poses an open engineering problem, due to highly variable object densities and bursty message load from object messages.

Therefore, Meru takes cues from the real world to aid in constructing a scalable virtual world system. In the next section, we discuss how mirroring and enhancing one constraint borrowed from the real world enables the Meru communication architecture to scale, even while providing sufficient service to users under worst-case conditions.

SCALABLE COMMUNICATION USING GEOMETRY

In the real world, physical limits constrain interactions between objects. While we can see and hear significant details about nearby objects, further objects lose detail and information. A virtual world renders to a two-dimensional display with finite information. This places a limit on the

amount of information a client needs to render the world. The same is true of audio: while there could be many sounds, an observer has a single audio stream. While nearby sounds in a stadium or city might be distinguishable, most simply become part of an underlying din.

Taking these analogies from space zero suggests that Meru spaces can exploit geometry to regulate communication in a virtual environment. Geometric information could be used to overcome one of the major limiting factors in modern virtual world systems: object density. A virtual world can be distributed across multiple servers and a geometric mapping exists from a region of space to a particular server. By limiting the communication rate between servers based on the size and proximity geometric regions they cover, the Meru architecture can bound the input traffic to every server.

In virtual worlds replete with objects maintaining perspective cameras revealing near objects at far greater sizes and higher resolutions than far objects, geometric distance serves as an interesting metric by which to control bandwidth flow. Using this metric, a space server allocates smaller portions of the available bandwidth to communicate with servers managing distant regions, and larger ones to servers managing nearby regions. This lets a space server receive higher-resolution data and more frequent updates from nearby regions as compared to regions that are farther away. One implication of this design is that objects wanting high-bandwidth communication must be close to one another in the world.

By maintaining an upper bound on the communication rate of each server and prioritizing the traffic according to geometry, we aim to achieve more scalability and efficiency than existing systems.

Benefits of rate limiting

There are several benefits of the rate-limiting approach described above:

- *Guaranteed bandwidth*. Under load, we can distribute bandwidth fairly which, assuming the underlying networking is reliable, guarantees some minimum bandwidth between two servers as a function of the virtual-world distance between spaces they host.
- *Straightforward scaling*. If demand for bandwidth exceeds supply, more bandwidth can be added by simply reducing the area of the world each server manages while adding more servers. Each server can still use the same maximum bandwidth.
- *Reduced need for over-provisioning*. The system can commit to providing some minimum quality-of-service between objects. A priori, we can determine the number of servers necessary to provide this service over a given world region. In times of low demand, we can dynamically reduce the number of active space servers, but will still know the maximum needed under load.

Geometric distance serves as an interesting metric by which to control bandwidth flow.

We believe that these benefits will greatly contribute to Meru being able to reach a planetary scale.

Calculating the communication rate

To benefit from these effects, Meru defines a *communication fall-off rate*, such that the bandwidth requirements from the entire virtual world *to* a server responsible for a particular region asymptotically approaches a constant, irrespective of the size of the world. The same would be correspondingly true for communication *from* that region. One such fall-off could be that the communication bandwidth between two objects falls off in proportion to the cube of their distances. This fall-off is intuitive for graphical systems, as distant portions of a scene can be sampled less frequently than nearby portions.[3] This implies that an observer can view a static virtual environment using a perspective camera obeying a bandwidth limit proportional to the observer's speed alone, irrespective of the size of the environment.

A space server can apply this logic to dynamic environments as long as updates to far pixels can be delayed proportionately to their distance from the viewer. This property holds true as long as all objects in the world obey a fixed speed limit, and it enables regions of the world to update each other only with bandwidth proportional to the inverse cube of their distances. Because graphics falls naturally into the bandwidth limit, the limit is promising for virtual world communication in general. We're also investigating the effects of bandwidth limits that fall off faster (such as Gaussians) to support more localized effects and those that fall off slower (such as the slightly superlinear functions of distance squared) for services with relaxed restrictions on rates of change.

The Meru project at Stanford seeks to explore how to build highly scalable 3D virtual worlds. Virtual worlds have been an anticipated medium for digital communication for a long time, but the systems of today fall short of their imagined potential. We believe that the key insight to making them achieve this potential is to model them after the physical world. By leveraging proximity, virtual and physical cross-reality, and using highly distributed services, we believe we will be able to build highly scalable virtual worlds. The real world is a comfortable metaphor for a wide range of issues that computer systems face today, such as security. Our hope is that not only would applying this metaphor lead to large, rich worlds, it will also make them easier to understand and accessible to use.

SPOTLIGHT

Daniel Horn is a PhD student in the Computer Science Department at Stanford University. Contact him at danielrh@cs.stanford.edu.

Ewen Cheslack-Postava is a PhD student in the Computer Science Department at Stanford University. Contact him at ewencp@cs.stanford.edu.

Tahir Azim is a PhD student in the Computer Science Department at Stanford University. Contact him at tazim@cs.stanford.ed.

Michael J. Freedman is an assistant professor of computer science at Princeton University. Contact him at mfreed@cs.princeton.edu.

Philip Levis is an assistant professor of Computer Science and Electrical Engineering at Stanford University. Contact him at pal@cs.stanford.edu.

ACKNOWLEDGMENTS

This work was supported by generous gifts from Microsoft Research, Intel Research, DoCoMo Capital, the National Science Foundation under grants #0831163 and #0831374 (NeTS-ANET), Stanford MediaX, and a Stanford Terman Fellowship. None of the work described in this article reflects the opinions or positions of any of these organizations.

REFERENCES

1. A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A Distributed Architecture for Online Multiplayer Games," *Proc. 3rd Conf. Networked Systems Design & Implementation* (NSDI 06), Usenix Assoc., 2006, pp. 155–168.
2. Akamai Technologies, Inc., www.akamai.com, 2009.
3. M. Regan and R. Pose, "Priority Rendering with a Virtual Reality Address Recalculation Pipeline," *Proc. 21st Ann. Conf. Computer Graphics and Interactive Techniques* (SIGGRAPH 94), ACM Press, 1994, pp. 155–162.