Large Scale VFX Pipelines

Matthew Chambers
Zorroa
mrc@zorroa.com

Justin Israel
Weta Digital Ltd.
justinisrael@gmail.com

Andy Wright
Weta Digital Ltd.
awright@wetafx.co.nz

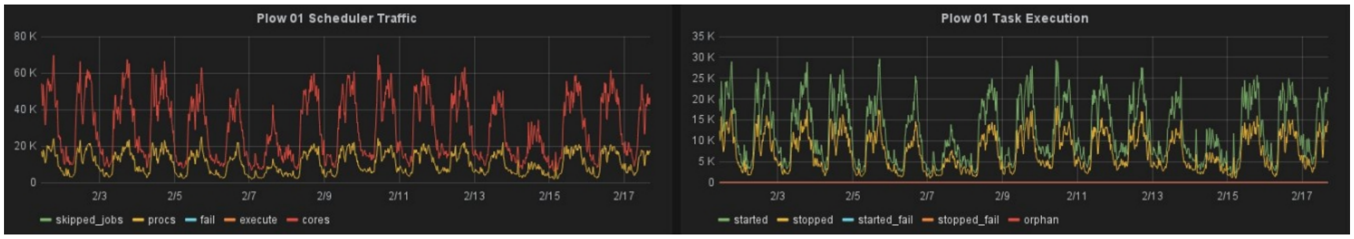

Figure 1: (left) Plow Scheduler Graph, (right) Plow Task Execution Graph

ABSTRACT

To ensure peak utilization of hardware resources, as well as handle the increasingly dynamic demands placed on its render farm infrastructure, WETA Digital developed custom queuing, scheduling, job description and submission systems - which work in concert to maximize the available cores across a large range of non-uniform task types.

The render farm is one of the most important, high traffic components of a modern VFX pipeline. Beyond the hardware itself a render farm requires careful management and maintenance to ensure it is operating at peak efficiency. In WETAs case this hardware consists of a mix of over 80,000 CPU cores and a number of GPU resources, and as this has grown it has introduced many interesting scalability challenges.

In this talk we aim to present our end-to-end solutions in the render farm space, from the structure of the resource and the inherent problems introduced at this scale, through the development of Plow - our management, queuing and monitoring software. Finally we will detail the deployment process and production benefits realized. Within each section we intend to present the scalability issues encountered, and detail our strategy, process and results in solving these problems. The ever increasing complexity and computational demands of modern VFX drives WETAs need to innovate in all areas, from surfacing, rendering and simulation but also to core pipeline infrastructure.

CCS CONCEPTS

• **Information systems** → Computing platforms; • **Applied computing** → Enterprise computing infrastructures;

KEYWORDS

distributed computing, pipeline, queuing, scheduling

ACM Reference format:
Matthew Chambers, Justin Israel, and Andy Wright. 2017. Large Scale VFX Pipelines. In *Proceedings of SIGGRAPH '17 Talks, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
https://doi.org/10.1145/3084363.3085021

1 OVERVIEW

WETAs hardware consists of 5 generations of blades, rackmount workstations, niche rendering hardware, alongside many generations of artist workstations. This lack of uniformity poses a significant challenge to any scheduling system, but add in the diversity of tasks required to run on this hardware and the problem becomes much more complex.

For many years WETA used Pixars Alfred for scheduling and dispatching, and over this time we have successfully adapted and extended it with custom priority systems to handle our available resource. A render farm expansion increased the resources significantly making it difficult to keep the machines fully utilized under this system. This coincided with Alfreds effective end of life - so an opportunity arose to explore other alternatives in this space. While resource utilization is a primary concern to a scheduler the suite of tools we had built around Alfred for queue management were significant, therefore any new system would require a robust management layer as well as a complete API for inspection and modification. Ultimately rather than utilize or customize an off-the-shelf product WETA chose to develop a custom solution from the ground up.

The resulting system - Plow - is built with an open API and provides rich statistical information that can be leveraged to rapidly build tools and graphs that help us drive our render farm effectively. A combination of ElasticSearch and Kibana lets us both manage the volume of data that Plow produces and rapidly produce visualizations to gain insight into the way our tasks ran. Graphite and Grafana provide graphing services allowing for historical views on tasks and help us detect trends in utilization over time.

2 PLOW - QUEUE MANAGEMENT, SCHEDULING, MONITORING

Plow is WETAs custom queue solution. It is a centralized task management system designed to maximize throughput and caters specifically to visual effects workflows.

2.1 Plow Server

When architecting a render queue management system one of the first goals is always scalability. In the past, achieving scalability meant separating the jobs into distinct processes which handled all of their own dispatching and communication with a central brain.

Another technique for scalability comprises of many separate processes or crond jobs that handle a small part of the workload. Over time, many distributed render queue systems can devolve into a sprawling array of processes and specialized databases running all over the facility. A recent trend in render queue architecture has been the movement from this distributed style architecture, to a centralized architecture where all job data is kept in a single data store. This can create its own issues however around performance and redundancy.

Plow is a hybrid of both the distributed and centralized model. It is built to run on a single SQL datastore, however it features clusterable business logic that can be used to provide performance, redundancy, and caching to any of its five major components: scheduling, dispatching, client API, render node API, and data maintenance. Each of these components has been designed as a microservice which runs within a single Plow server process, and each of these micro-services can be independently tuned, enabled or disabled. This architecture allows for specialized nodes which can be uniquely configured to handle a particular task or type of traffic. Large Plow deployments, such as that used to manage WETAs onsite render farm, run nodes specifically for handling scheduling and dispatching, while another set of nodes handle the massive amount of API calls generated by the client tools and libraries. All of these nodes work together as one large clustered application to provide a scalable and redundant queueing system for our pipeline. Smaller, single node Plow deployments are able to be setup to run on-set and have even been used remotely on-location in the forests around Vancouver. The single process design of Plow makes it easy for administrators to get Plow up and running anywhere to handle any scale of resource.

2.2 Plow Render Node Daemon

A standard component to any render queue solution is a running daemon process on each worker host, allowing tasks to be started on behalf of the scheduler and the submitting user. A well designed daemon would ideally have some of the following qualities: Being low impact on the host, so as not to waste resources that could be used for actual tasks; Concurrent handling of task scheduling communications; Accurate metric reporting for active and completed tasks; Multi-platform support for improved deployment portability and flexibility; Extra smarts surrounding the reservation of resources to each scheduled task.

Plows Render Node Daemon (rndaemon) is a custom solution, with modular multi-platform support for Linux, OSX, and Windows. Linux support makes heavy use of Control Group containers (cgroups) in order to jail each scheduled task to the exact and ideal set of logical cores that have been allocated. Cgroups also provide fine-grained control over processes management, and highly accurate metrics collection. rndaemon also has a sizeable RPC API for control by both the Plow Server, and integration with the Plow Desktop Control UI application, allowing users to opt their machines into the render farm as various levels of commitment.

2.3 Plow Client API and Frontend

Having a strong render queue server (scheduling, dispatching, process managers) is only half the story of a complete solution. Rich user-facing tooling is desirable providing clients with a transparent view onto the state of their jobs, allowing them to easily build reusable tools specific to their pipelines and workflows, as well as allowing for extension of the core toolset with custom logic. Plows RPC client API is built on top of Apache Thrift, allowing bindings to be generated for multiple programming languages. Official client APIs exist for Java, C++, and Python, and some departments have built frontend web tools, leveraging the JavaScript AJAX generated API.

The client API is very comprehensive and high level, which means that 99% of the operations that one could do in the plow-artist man- agement GUI are directly available via the API. Plows client APIs are fault-tolerant, load-balancing, thread-safe, and ensure that ac- cess patterns are made through an approved public interface written by the core developers.

Alongside the client API, Plow provides a modern and powerful suite of GUI tools. plow-artist and plow-wrangler are panel-based (modular) GUIs for viewing and managing the state of the render farm. Different panels, and panels grouped into custom Workflow layouts, provide different views and tools catering to Artists, Wranglers, and Production. The GUIs also offer various hooks for customizations on a per-user or per-pipeline basis. These hooks include custom log viewers, custom context-aware actions discovered at runtime, custom media viewers associated with file types, and the integration of arbitrary attributes on Jobs and Layers with view functionality. plow-desktop-control is a system tray GUI allowing users to conditionally opt their workstations into the render farm resource pool. Users can participate in tasks for specific projects or make their machine available to the whole render farm, allocating as little or as much memory and cpu as they wish.

3 PRODUCTION IMPACT

Deploying a large scale change to core infrastructure at a facility that sees infrequent downtime is no easy task. Kenobi - our job description framework - allowed existing pipelines to migrate to Plow with minimal risk to production. The flexibility and scalability of the new systems, along with a demonstrated ability to successfully process complex jobs that had not been previously possible, motivated key stakeholders to promote adoption across departments.

With Plow deployed WETA saw a number of benefits, from improved resource utilization to better management and monitoring tools for artists. A much richer set of statistics allowed WETA to spot inefficiencies, optimize job descriptions and continually tune the render farm infrastructure.