Blender Distributed Rendering

Blender Conference 2013, Amsterdam

James Yonan

Cloud rendering economics

- Build your own render farm using physical hardware vs. rendering in the cloud
- AWS 8-core VM rents for US$0.07 per hour at the lowest spot market price (c2.xlarge
- Purchase of comparable hardware estimated cost: US$125/core x 8 cores = US$1000.
- Cost of electricity, assuming US$0.10 per kilowatt hour and 200 watts energy consumption: US$0.02 per hour.

Cloud rendering cost comparison

- Cloud instance is $0.05/hour after adjusting for electricity cost
- $1000/$0.05 = 20,000 instance hours = ~ 2.25 years of 24x7 usage

Scalability

- Comparing cloud costs vs. bare metal misses a key point.
- Fundamentally, the cost of renting 1000 hours of computer time in the cloud is the same whether we rent one instance for 1000 hours or 1000 instances for one hour.
- With the cloud, we can throw massive amounts of computer power at our render job, and at the end of the day, the costs are essentially the same as if we took the slow boat.

Parallelizing Blender Rendering

- Rendering animation frames is an inherently parallel operation
- With a simple command, we can tell blender to render a specific frame of a project, e.g.
- $ blender -b proj.blend -F PNG -o frames/frame_#### -s 5 -e 5 -j 1 -t 0 -a
- We can use this capability of Blender as a building block to construct a render farm.

Brenda – Blender render farm for Amazon Web Services

- I've developed a set of Python scripts that make it fairly straightforward to use Amazon Web Services (AWS) as a Blender render farm.
- I've published these scripts on github: https://github.com/jamesyonan/brenda
- A Brenda tutorial is provided in the project README file.

The Amazon EC2 cloud

- Amazon has created a marketplace where computers can be rented by the hour.
- In addition, Amazon has created a spot market where the cost of computer rental is determined by market supply and demand.
- This is an excellent development for the problem of distributed rendering, because we can render during off-peak periods when the cost is low.

Spot market render challenges

- Rendering in the EC2 spot market creates some implementation challenges for render farm software.
- While the cost is low, EC2 spot instances can be killed at any time if the market price of the instance rises above our maximum bid price.
- Therefore, our render farm software must be fault-tolerant against instances being created and killed at random during the course of the render.

SQS and Fault tolerance

- Amazon provides a service called SQS (Simple Queue Service).
- Like most queues, SQS allows us to create and consume a work queue across the cloud.
- SQS also includes a system for acknowledgment of pending tasks, so that if a render node is killed before a task is complete, its pending tasks will be returned to the work queue.
- Brenda leverages on SQS by using it as a staging system.

Brenda is optimized to use the lower-cost spot market

- Leverages on SQS for fault-tolerance, so that render work queue is seamlessly processed even as render nodes are created and destroyed at random.
- Allows render instances to be flagged as “persistent” so that they will be automatically restarted after spot market price spikes.
- Allows subframe rendering, to minimize lost work when instances are killed with pending render operations.

Fundamentals of Brenda

- Push a Blender project to the S3 file store.
- Create a work queue that describes blender commands to render specific frames.
- Unleash the AWS cloud on the work queue.
- Each node in the render farm takes a task out of the work queue, renders it, and pushes the render product (such as PNG files) to an S3 file store.

Brenda workflow