Multiverse: a Next Generation Data Storage for Alembic

Aghiles Kheffache*
The 3Delight Team / J Cube Inc.

Marco Pantaleoni, Bo Zhou†
J Cube Inc.

Paolo Berto Durante‡
Polygon Pictures Inc. / J Cube Inc.

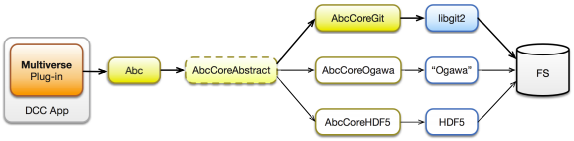

Figure 1: *multiverse architecture diagram. This poster describes our work underneath the Abc abstraction layer.*

1 Abstract

We introduce *Multiverse*, an open source[1] next generation data back-end to the widely used *Alembic* file format. Our back-end relies on *Git*, a powerful distributed source control system. We inherit all the features introduced by *Git*, including: compact history and branching, natural data de-duplication, cryptographic data integrity, SSH internet sharing protocol and collaborative work capabilities. Our scene data representation allows for punctual access to individual scene elements, opening the door to multi-threaded I/O as well as easy scene updates. To our knowledge, it is the first time that such a set of features is available to the production community.

2 Overview

Alembic [ILM & SPI 2009] is a data representation scheme for storing graphics scenes. It is widely used for both scene exchange and to encapsulate procedurally accessible geometry. Alembic relies on both HDF5 [The HDF Group 2000] and "Ogawa" back-ends to store data. While these file formats provide good performance and functionality, they are closed to advanced features such as versioning, branching and collaborative work.

Git has become the most widely used source control system. *Git* data (or "blobs") are addressed by their SHA-1 hashes, achieving natural data de-duplication. This aspect is of particular interest for scene storage since geometry duplication is common, both inside one frame and across multiple frames in an animation. In addition to compact data representation, *Git* allows for efficient *history* representation. Most importantly, the same data de-duplication mechanism is extended through "revisions" so to share data across scene's multiple versions. Extremely interesting features such as cryptographic data integrity are unfortunately out of the scope of this poster.

3 Implementation

Thanks to a well defined back-end API in Alembic, we wrote a plug-in that is API compatible with added functionality for history management[2]. We now describe how we use *Git* to store data and what "data view" model we use to access 3D scene's hierarchy.

Data View Model To mirror Alembic scene representation, we use a *virtual* directory hierarchy on disk. In this representation, geometry and attributes are stored as files at leaf nodes while hierarchy is expressed as directories. We store geometry in binary for compactness. Attributes are stored as JSON files for ease of access and manipulation. Note that this data view is *virtualized*: it is not visible to the user unless a "checkout" of the *Git* repository is performed. Such checkout operations can be performed on individual elements of the scene hierarchy to perform manual or scripted modifications — a very handy feature in a production environment.

Data Structure On Disk We rely on LIBGIT2 [libgit2 2009] to read and write *Git*-based scene repositories and to virtualize the data view model described above. The library gives direct access to the repository without going through a "checkout" of the scene description. In other words, we directly write tree, blob and commit objects which are the fundamental building blocks of a *Git* repository. As a first prototype implementation, we tried to write non-virtualized scene structure to disk, but this led to unmanageable performance. Note that *Git* stores data in a directory structure of its own, but this is not to be confused with our data view model.

4 Results

We timed *Multiverse* against Alembic's standard HDF5 and "Ogawa" back-ends with different production scenes. Results show much improved overall data size, a performance degradation in reading (which is however fast enough for production usage) and in writing especially on Windows. This was expected since data access speed is now tied to file system directory access performance (unlike HDF5 and "Ogawa", *Git* stores data in a directory structure). Note however that data size and performance isn't adversely affected by the presence of many scene versions in one repository.

Table 1: *Performance overview*

Scene	Complexity	Disk Size	Write	Read *(sec)*
Sidonia 1	medium	½	2x	3s
Sidonia 2	high	½	3x	12s
Transformer	very high	½	4x	40s
+History (10 edits)	any	½↔⅑	≈3x	edit-dep.

5 Conclusion and Future Work

We introduced a powerful, open source, next generation, data back-end for Alembic as well as a set of new features that are yet to be fully explored by production communities. The current implementation has a much more compact data representation than HDF5 and "Ogawa" but still lacks in I/O performance, especially when writing. This will be the focus of future work. Leveraging multi-threading of LIBGIT2 is another area of interest.

References

ILM & SPI. 2009. Alembic format description.

LIBGIT2. 2009. libgit2 description.

SCOTT, C. 2009. Pro git 1st ed.

THE HDF GROUP. 2000. Hierarchical data format version 5.

*aghiles@{3delight.com, j-cube.jp}
†{marco, bo}@j-cube.jp
‡paolo@ppi.co.jp

[1] `http://multi-verse.io`

[2] Such as specifying commit messages and selecting revisions.