Tangible Performance Management of Grid-based Laptop Orchestras

Stephen David Beck
Louisiana State University
Baton Rouge, Louisiana
sdbeck@lsu.edu

Chris Branton
Louisiana State University
Baton Rouge, Louisiana
branton@lsu.edu

Sharath Maddineni
Louisiana State University
Baton Rouge, Louisiana
smaddineni@cct.lsu.edu

ABSTRACT

Laptop Orchestras (LOs) have recently become a very popular mode of musical expression. They engage groups of performers to use ordinary laptop computers as instruments and sound sources in the performance of specially created music software. Perhaps the biggest challenge for LOs is the distribution, management and control of software across heterogeneous collections of networked computers. Software must be stored and distributed from a central repository, but launched on individual laptops immediately before performance. The GRENDL project leverages proven grid computing frameworks and approaches the Laptop Orchestra as a distributed computing platform for interactive computer music. This allows us to readily distribute software to each laptop in the orchestra depending on the laptop's internal configuration, its role in the composition, and the player assigned to that computer. Using the SAGA framework, GRENDL is able to distribute software and manage system and application environments for each composition. Our latest version includes tangible control of the GRENDL environment for a more natural and familiar user experience.

Keywords

laptop orchestra, tangible interaction, grid computing

1. INTRODUCTION

Laptop orchestras[8] (LOs) use an orchestral metaphor to provide an engaging and challenging environment to experiment with human-computer interaction, network and machine latency, and sound/signal processing. LO performers use ordinary laptop computers as instruments and sound sources for performing specially created compositions[7]. With the recent successes of the Princeton and Stanford laptop orchestras, LOs have now been established at many universities in the US, the UK, and as private ensembles around the world[11, 13].

LO composer-performers develop software to interpret human actions through computer interfaces that in turn control virtual instruments and processes that ultimately render music. Compositions can be improvised or scored, of determined or indeterminate length, with or without acoustic musicians. Laptops communicate across WiFi networks to synchronize time, distribute control messages, and manage other performance information.

Distribution, management and control of the necessary software across a heterogeneous collection of networked devices is a tremendous challenge for LOs. Each LO composition describes a potentially unique combination of core software, middleware and user-interface software that must be initialized, launched and performed. Configuration can range from the very simple (e.g., a single program on each machine, responding to keyboard events), to the very complex (Wii-motes, iPads, custom UI and laptops, driven by a networked time-sync). Laptops may all behave the same, or play specialized roles. The complexity of each piece and skill of each performer can affect the amount of time needed to prepare a piece for performance. And the "performance complexity" can scale exponentially with the number of laptops in the ensemble. Software may be stored in a central repository and distributed before a concert begins, but individual laptops must be configured and initialized immediately before the performance of each piece. Princeton's laptop orchestra identified software configuration as one of their most significant problems[9].

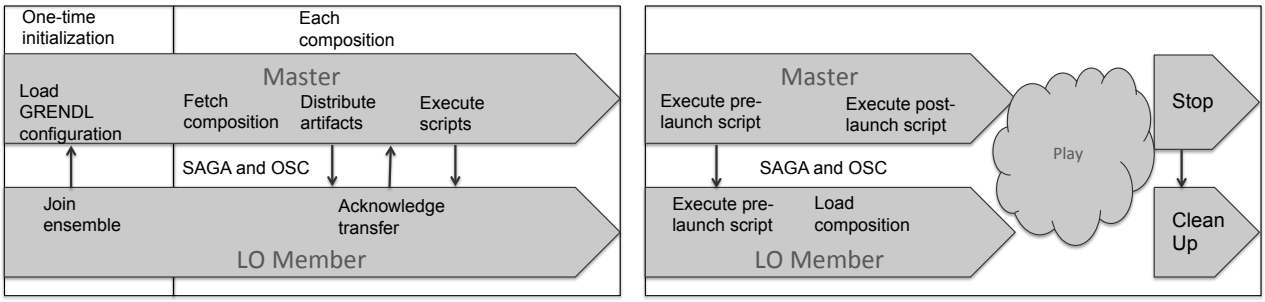
Our group has developed and field-tested the GRid ENabled Deployment for Laptop orchestras (GRENDL) system to help address the challenge of managing LO software distribution and configuration, while providing an experience for ensemble members that closely mimics that of a conventional orchestra. GRENDL is an integrated system that deploys, manages, and controls software and hardware technologies needed for the performance of music for laptop orchestras. For a given LO composition, GRENDL links digital artifacts (e.g., scores, software, electronic devices) with middleware applications (e.g., ChucK[12], Max[5], SuperCollider[4]) specific to the devices and operating systems available for performances.

2. LO PERFORMANCE WORKFLOW

A primary aim of GRENDL development is to support a workflow that is familiar to musicians. To maintain the orchestral metaphor, the system recognizes two distinct classes of ensemble computers. A single *master* machine, which is usually but not necessarily associated with the conductor, is responsible for loading and distributing the compositions, beginning and ending each piece, and managing the order of play. Any number of *performer* machines can join the ensemble, and play a specific role (i.e., instrument and part) in a specific piece.

The functionality provided by GRENDL can be seen as roughly analogous to the music librarian of a conventional orchestra, retrieving the parts for each musician and distributing them to the proper workstations. Some additional complexity is introduced in the LO case, since each laptop can (and likely will) serve as a different instrument for each

NIME'11, 30 May–1 June 2011 Oslo, Norway



(a) GRENDL transfer process

(b) GRENDL launch process

Figure 1: GRENDL supports a natural workflow between conductor (master computer) and orchestra members. (a) Before the performance, GRENDL loads the program and initializes the list of ensemble members. Digital artifacts are then transferred to each machine, ready for launch. (b) During the performance, GRENDL launches scripts to initialize each performer workstation, launch and configure the necessary middleware, and load the necessary data files. When the piece concludes, GRENDL launches scripts to restore each performer laptop to its default configuration.

composition. GRENDL distributes the appropriate music to LO musicians for a concert before the performance begins. GRENDL can also provide the conductor with the correct scores for each piece on the program, and give the conductor the ability to start and stop each piece in turn.

The entire program is transferred before the performance to minimize the delay between pieces, though the GRENDL architecture can support transfers at any time that a piece is not being played. The conductor can initiate a transfer command directly to the GRENDL engine using the command-line interface, or utilize the recently developed GRENDL Conductor application to manage the performance.

During the concert, the conductor instructs GRENDL to launch the software for each piece. GRENDL executes pre-launch scripts on the master computer to communicate roles, synchronize the ensemble, or perform other custom initializations, sends a "start" command to all computers in the orchestra, and runs post-launch scripts (if needed) on the master. Ensemble members use the GRENDL Performer interface to signal the conductor when their machine is properly configured and ready to play.

Once each laptop is configured and the proper software is launched, the piece is played. At the conclusion of each piece, the conductor triggers GRENDL's "quit" mode, which executes cleanup scripts on the machines.

The number of different scripts, configurations, hostnames, and other technical details that are needed to configure a LO can create a heavy cognitive burden on performers. "Cartouche" tangible user interface tokens[10] provide a simple and convenient way to encapsulate digital content and operations. GRENDL is designed to use cartouches to link client computers with specific performers and roles (e.g., percussion, voice). Cartouches may also be used to trigger software actions on the client computers, initiate messages to the GRENDL Conductor, or be linked to performance graphics for the musicians to use as a score.

3. GRENDL ARCHITECTURE

GRENDL has been created specifically to address the challenge of distributing and configuring software for LO performances. This is accomplished by viewing a LO as a computational grid [1]. One master machine, normally the conductor, assigns "jobs" to each of the remote performer nodes. In the case of GRENDL, the "transfer" jobs consist of using one of a variety of network protocols to distribute scripts, patches, program files, and other digital artifacts needed to play a composition. The "launch" jobs instruct the performer laptops to execute a series of scripts that configure the machine to play a specific composition. "Quit" jobs execute another script that stops any running software and cleans up any changes that were made to the environment.

The GRENDL software architecture includes components to manage connections to performer latops, retrieve and distribute the compositions, configure middleware and (when necessary) make system-level configuration changes, launch each composition, and restore the laptop meta-instruments to their default configurations after the piece has been performed. A conceptual overview is shown in Figure 2.

At the heart of the system is the GRENDL engine, a command line program written in C++ that is responsible for distributing and managing the jobs of each computer. The engine manages the transfer, launch, and quit jobs according to parameters specified in a set of configuration files associated with each piece in the program. The

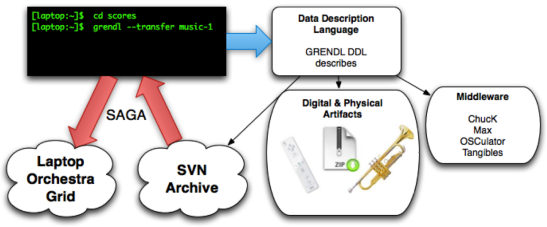


Figure 2: GRENDL includes components to retrieve compositions from online archives and distribute them to LO members. Other components load and configure middleware applications and compositions according to GRENDL data description language specifications.

current implementation of GRENDL uses the Simple API for Grid Applications (SAGA)[2] to manage the LO grid. SAGA is a grid computing framework that helps manage distributed applications in complex environments. SAGA connects application software written in C, C++ or Python with grid-based middleware services for distributed computing, providing a robust and platform neutral environment for complex computation. While SAGA was developed for high performance scientific applications using large grids of

hundreds or thousands of computers, it has adapted readily to the laptop orchestra environment. GRENDL leverages the SAGA framework for the distribution, initialization and launch control of LO software, treating the LO as a unique application of grid computing for live music performance.

An implementation of SAGA forms the file management and remote execution core of GRENDL, including support for configuring laptops, distributing compositions and supporting software, and launching the environment for each piece in the program. Since SAGA's synchronization is event-driven rather than time-based, it provides an ideal infrastructure for the asychronous and variable latency activities typical of performance preparation. These same features limit SAGA's utility during the performance, though SAGA still provides an important mechanism for file transfer and middleware configuration between compositions.

The most recent version of GRENDL uses Open Sound Control (OSC)[14], a protocol for communication among computers, sound synthesizers, and other multimedia devices, for more immediate and lightweight network communications. OSC is a simple, powerful protocol that provides everything needed for interactive control of sound and other media processing while remaining flexible and easy to implement. OSC libraries exist for most major programming languages, including C/C++, Java, Max[15] , and ChucK[12]. OSC interfaces have also been developed for a large number of interaction devices and visualization systems, as well as the majority of electronic instruments.

GRENDL uses OSC as the primary mechanism for intermachine communication during performances. This includes communication between the Conductor and Player applications, as well as distributing events generated by the tangible controls. OSC-based applications developed for the iPhone/iPad platform allow the conductor to set parameters for the ensemble and exchange information with performers. Max-based GUI's make use of OSC for communication between performers, and several ChucK pieces utilize OSC for synchronization and timing.

4. GRENDL CONDUCTOR

Conductor provides overall performance management for GRENDL. Conductor communicates workflow events to the members of the ensemble and manages the transitions from one piece to the next. It is the Conductor component that makes the calls to the GRENDL engine that will in turn deliver the jobs to the performers' laptops.

On startup, Conductor loads the program for the upcoming performance. Each item in the program represents a piece in the performance, including the location of the composition's digital artifacts. This location may be a folder accessible to the master computer, or the URL of an online repository.

Along with the program, Conductor loads a description of the ensemble, including account names and network addresses for all members. Before transferring the compositions to the laptops, Conductor listens for new members to join, and allows other members to be removed. Once the ensemble is complete, Conductor instructs the GRENDL engine to transfer each composition in the program to each orchestra member. After transferring all of the files, Conductor transitions to performance mode. By default, compositions are played in the order in which they are listed in the program, though this order can be changed through the user interface. For each piece, Conductor instructs the GRENDL engine to assign the appropriate "launch" job to each member of the ensemble. Once the ensemble is ready, Conductor enters "playing" mode. Until the signal is given to stop playing, Conductor will not send any other signals or process any remote requests. This minimizes the possibility that GRENDL will interfere with the performance of the piece.

The initial Conductor user interface was developed in Processing [6] to explore the capabilities needed to manage LO performance using GRENDL. It is expected that most or all of the interaction capability in Conductor will eventually be realized with a tangible user interface.

5. GRENDL PERFORMER

Complementing the GRENDL Conductor component is the Performer application, which is deployed on each orchestra member's laptop. Like Conductor, Performer was developed in Processing.

Performer provides an endpoint for communication between ensemble members and the Conductor. This allows ensemble members to register their laptops with the Conductor, thereby adding the machines to the laptop grid. Performer informs members of state changes in the performance, such as transfer and launch of specific pieces, and it allows orchestra members to signal the Conductor of specific events, such as when they are ready to play.

In addition to an OSC server, Performer monitors serial communications to detect events from the RFID cartouche readers. These events are translated into OSC messages and transmitted to the Conductor, except when a piece is being played. As with Conductor, Performer does not send events while a piece is being performed, and will only respond to the "quit" job sent from the Conductor.

6. TANGIBLE CONTROL OF GRENDL

The number of separate operations that must be performed to prepare to play a piece in a LO can be daunting. Configuring audio channels, loading middleware and data files, connecting and configuring external tools (e.g. Wiimote), and synchronizing multiple machines takes time, all while an audience watches. GRENDL helps address this problem, but at the cost of another layer of configuration, and another set of commands to be remembered.

Tangible user interfaces provide an effective answer to this new challenge. Specifically, cartouche tangibles[10] provide convenient tokens to represent concepts or actions. Cartouches can provide legible and actionable representations of compositions, performers, instruments, and programs that are usable by both human performers and electronic components of the LO. Cartouches provide a natural way for performers to interact with GRENDL, as well as a convenient and tangible representation of the elements of electronic music.

Cartouches have been tested with GRENDL in two contexts. First, a component has been developed that uses the Trackmate computer-vision based fiducial tracking system[3] to help configure the ensemble. When fully integrated, this will enable rapid and reliable reconfiguration of the orchestra for different pieces. The Trackmate system (Figure 3b) tracks the presence, position, and rotation of each cartouche, making a number of parameters available for future use.

LO members can be equipped with RFID tagged cartouches that are linked to the performer's identity within the group, machine information, or their specific role in a composition. In addition to concepts or entities, cartouches may represent actions or states, such as "ready to play."

7. CONCLUSIONS

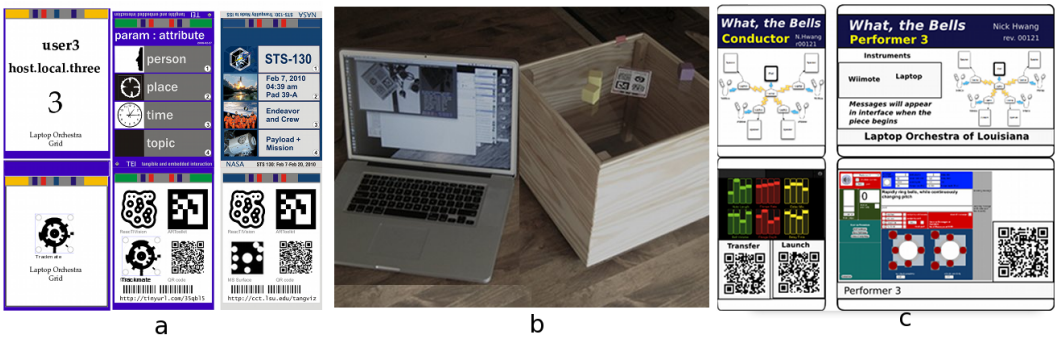


Figure 3: (a) Cartouche tangibles may represent a single concept, event, or role, or an entire set of values; (b) Trackmate is one of several computer vision systems that can provide a convenient way for LO members to interact with cartouches; (c) Cartouches are designed to be legible to humans as well as computers.

Our initial tests with the command-line version of GRENDL have demonstrated that it can be used successfully in a concert environment. It was initially piloted during a performance of the Laptop Orchestra of Louisiana (the LOLs) on April 14, 2010. Here, GRENDL was used to manage two LO compositions, and worked without incident. Over the following year, GRENDL was further tested on a regional tour by the LOLs, and managed an entire concert on April 4, 2011. We found that writing configuration settings was not nearly as straight forward as we wanted. Once tested and configured correctly, GRENDL worked flawlessly, and facilitated a very successful concert without any delays during the performance. We anticipate that the tangible version of GRENDL, which we will use in the coming concert season, will alleviate many of these issues.

These experiences have proven GRENDL's utility, confirming our belief that such a system addresses key impediments to the widespread adoption and long-term persistence of the laptop orchestra genre. That said, these tests have revealed additional parameters and actions that must be considered when building transfer, launch and quit scripts, especially in the realm of complex OS configuration.

Extending the GRENDL engine to integrate smoothly with tangibles, and in novel runtime environments, will require extensions to SAGA. The trans-disciplinary nature of GRENDL provides potential to shed new light on existing challenges in computational science. The LO setting presents a unique perspective from which to investigate topics such as time-sensitive and dynamic job scheduling, latency-bound interaction, and effective user interfaces for grid computing environments. Some of the first iteration interaction technologies have been developed for distributed computational science applications, and some of what is learned through GRENDL will likely be applicable in that area.

8. ADDITIONAL AUTHORS

Additional Authors: Brygg Ullmer (Louisiana State University, email: `ullmer@cct.lsu.edu`) and Shantenu Jha (Louisiana State University, email: `sjha@cct.lsu.edu`).

9. REFERENCES

[1] F. Berman, G. Fox, and A. Hey. *Grid Computing: making the global infrastructure a reality*. John Wiley & Sons Inc, 2003.

[2] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. A simple API for Grid applications (SAGA). In *Grid Forum Document GFD*, volume 90, 2007.

[3] A. Kumpf. *Trackmate: Large-scale accessibility of tangible user interfaces*. PhD thesis, Massachusetts Institute of Technology, 2009.

[4] J. McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68, 2002.

[5] M. Puckette. Max at seventeen. *Computer Music Journal*, 26(4):31–43, 2002.

[6] C. Reas and B. Fry. Processing: a learning environment for creating interactive Web graphics. In *ACM SIGGRAPH 2003 Web Graphics*, page 1. ACM, 2003.

[7] S. Smallwood, P. Cook, D. Trueman, and G. Wang. Composing for laptop orchestra. *Computer Music Journal*, 32(1):9–25, 2008.

[8] D. Trueman. Why a laptop orchestra? *Organised Sound*, 12(02):171–179, 2007.

[9] D. Trueman, P. Cook, S. Smallwood, and G. Wang. Plork: Princeton laptop orchestra, year 1. In *Proceedings of the 2006 International Computer Music Conference*, pages 443–450. Citeseer, 2006.

[10] B. Ullmer, Z. Dever, R. Sankaran, C. Toole Jr, C. Freeman, B. Cassady, C. Wiley, M. Diabi, A. Wallace Jr, M. DeLatin, et al. Cartouche: conventions for tangibles bridging diverse interactive systems. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 93–100. ACM, 2010.

[11] G. Wang, N. Bryan, J. Oh, and R. Hamilton. Stanford Laptop Orchestra (SLOrk). *Proceedings of the International Computer Music Conference*, pages 505–508, 2009.

[12] G. Wang, P. Cook, et al. ChucK: A concurrent, on-the-fly audio programming language. In *Proceedings of International Computer Music Conference*, pages 219–226. Citeseer, 2003.

[13] G. Wang, D. Trueman, S. Smallwood, and P. Cook. The laptop orchestra as classroom. *Computer Music Journal*, 32(1):26–37, 2008.

[14] M. Wright and A. Freed. Open sound control: A new protocol for communicating with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference*, pages 101–104, 1997.

[15] M. Wright, A. Freed, and A. Momeni. Opensound control: State of the art 2003. In *Proceedings of the 2003 conference on New Interfaces for Musical Expression*, page 160. National University of Singapore, 2003.