Teaching HPC Concepts with Serious Games

Lauren Milechin
MIT
Cambridge, MA
lauren.milechin@mit.edu

Julia Mullen
MIT Lincoln Laboratory
Lexington, MA
jsm@ll.mit.edu

***Abstract*—This Innovative Practices Work in Progress describes how serious games can be used to help teach High Performance Computing (HPC). Serious games provide pathways for learners to develop intuition while learning new concepts. We define serious games as those that incorporate learning objectives, educational content, and assessment and whose purpose is learning rather than entertaining. Such games are valuable in an educational context because they engage students in active learning and help students develop mental models from their experiences. These student experiences generally lead to deeper questions, allowing instructors the chance to clarify misunderstandings, and reinforce learning. This work describes two games used in informal HPC courses to provide students with tangible hands-on experiences with HPC concepts.**

I. Introduction

High Performance Computing (HPC) education and training is rarely integrated into formal academic curricula and courses, yet to be successful students and professionals need to understand how to effectively use and scale research codes and software packages in order to develop insight into large, complex systems. While significant resources exist for training, including several online repositories supported by professional organizations and HPC consortia in the United States, United Kingdom, and European Union, [1]–[4], a common thread in these resources is the tendency to either be very abstract or dive too deeply into details, including details that are specific to a given HPC System.

To prepare researchers to use a local, interactive supercomputing system, the authors provide training through informal workshops or short non-credit courses. These workshops include an overview of supercomputing concepts and parallel programming paradigms as well as more concrete skill-based content illustrating the specific details required to use local HPC centers. The participants are predominantly researchers and graduate students from non-Computer Science STEM fields. These students have deep knowledge of their discipline but basic computational skills and minimal exposure to shared Linux systems. To engage this population, there are no prerequisites for the workshop, beyond having a research project that requires supercomputing resources and knowledge of one programming language.

Even with good learning design, the compressed time frame associated with workshops limits opportunities to revisit and reflect on the material, activities that students regularly engage in during term or semester based courses. The result is that learners, especially novices, develop a superficial understanding of the discipline and often grasp the first HPC strategy presented to them and are later unable to scale their application beyond the most basic implementation. To overcome the limitations of time and repetition, the authors theorize that using serious games in HPC education and training could increase student intuition around parallel and distributed computing. In this context, serious games are used as a means to provide learners with exposure to HPC systems and architectural components in a manner that allows them to explore the behavior, opportunities, and limitations that impact time to solution for their research applications. Furthermore, this use of serious games provides a venue for both greater discussion and repetition, allowing students a chance to build understanding at their pace [5]. It is important to note that this paper focuses on using games to teach concepts rather than using the creation of games as a programming objective. Furthermore, the focus is not the inclusion of gamification elements in educational activities, e.g. badges or leaderboards, despite evidence indicating that gamification leads to greater short term knowledge retention even for workshops [6].

The use of games in education has a long history and the term "serious game" was first used by Abt [7] to describe games designed for learning. Serious games for military, medical, and business education are rapidly growing domains [8]. In 2017 Petri et al. [9] analyzed 114 educational games for computer science, for a target audience extending from grammar school through university [10]. For our audience, the goal is to help STEM (non-computing majors) learners develop the HPC competencies outlined in the redesign of ACM/IEEE computer curricula and described in the ITiCSE working group report on HPC Education [11]. This motivated the creation of games desinged to develop intuition about HPC competencies in an active, participatory format.

The games discussed here were developed for a physical classroom setting and then modified for synchronous Zoom

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001.Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

classes. The original decision to develop games using tangible components for in-person activities rather than digital games came from the desire to remove abstractions that often obscure concepts for the novice learner. In the games discussed in this paper, the instructors used the following framework outlined by DeGloria et. al [12],

- set the stage with a set of key questions that directly correspond to the learning goals,
- provide a brief description of the game rules and goals, and
- conclude with a discussion of the key questions to assess and extend learner understanding while also clarifying any misunderstanding.

This paper presents the games, the modifications incorporated to improve game flow, and some lessons learned. As this is a work in progress, we have yet to conduct a formal evaluation of these games. Determining the best metrics for evaluation and collecting evaluation data is the next step in this research project.

II. Example 1: Be the Scheduler

The "Be the Scheduler" game allows students to take on the role of the scheduler in an HPC system. This game was inspired by the question that HPC users often ask: "Why isn't my job running?". It was developed for a class aimed at current and near-future users of the local HPC Systems. These systems tend to have very heterogeneous job types and workloads, ranging from high throughput to tightly coupled MPI jobs, and we wanted a game that would demonstrate how the scheduler handles diverse resource requirements.

In the game, Lego Bricks represent the cluster, nodes, and different types of jobs, and cards represent submission scripts. Students draw cards and decide where the jobs should run and launch them by placing the job bricks onto the nodes. The goal of this game is to schedule and run all the jobs in the job submission card deck. The learning objectives are to understand the role of the scheduler, develop an intuition about how resources are allocated, understand the range of job types and resources required by each, and show the difference between distributed and shared memory applications.

A. Game Play

To set up the game, flat Lego bricks, nodes, are placed on a larger flat board, the cluster. The size of a brick corresponds to the number of cores on that node. Memory is not directly represented, instead each core on the node is allotted an equal portion of the total memory, mimicking the system that the game is modeled after. There are two node types: CPU and GPU nodes. GPU nodes have flat 1×1 bricks placed adjacent to the node to represent GPUs. The board is placed on a table with space for two lines of cards below it: the Running and Pending queues. The Running Queue contains the cards for currently running jobs, and the Pending Queue contains jobs that are waiting to run. The job card deck is placed to the side of the board. Figure 1 shows some sample job cards. The front side of each card, displayed when the job is pending, shows

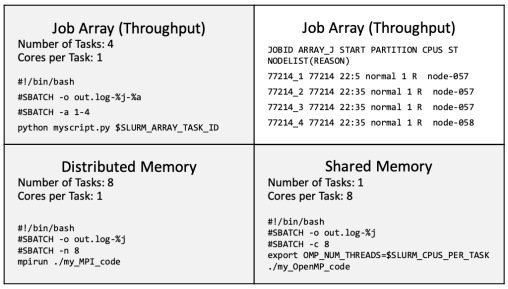

Fig. 1: Sample job cards. The front of the cards are gray-toned to differentiate from the back.

the type of job, the resources required, and a short submission script. The back of each card shows the job status output.

During game setup, six cards are drawn and their requested jobs pieces are selected and launched. Students launch a job by placing the bricks corresponding to that job on the board, and then placing the card face down in the Running Queue so the job status can be seen. Lego bricks of different colors and shapes represent different types of jobs, and are described below. This setup can be done collaboratively and does not have to be done in turns.

The game proceeds by taking turns going through the job card deck until all jobs have started. Figure 2 shows a board during play with jobs running on the cluster and waiting in the Pending Queue. For each turn the student rolls a die to determine the completed job (the number on the die corresponds to the job that has completed), removes the job from the board and its corresponding card from the Running Queue, and launches pending jobs. To do so, they first draw a new job card from the deck, place it at the end of the Pending Queue, and place the corresponding job pieces on the card. Then, starting at the beginning of the Pending Queue, they launch any jobs that can run. As jobs start, the player places the card at the end of the Running Queue face-down, and places the job pieces on the board. There are five job types:

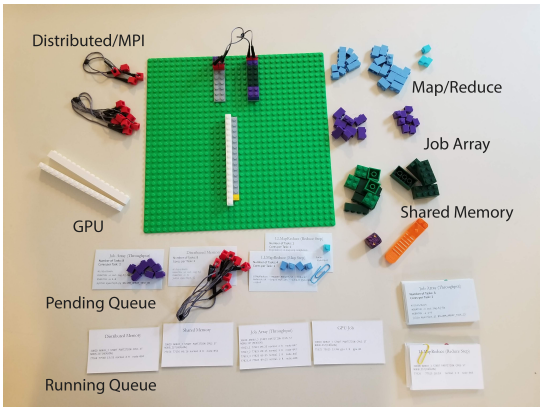

Fig. 2: Layout mid-game. The board shows four jobs running. Game pieces are shown to the sides of the game board, with queues at the bottom.

- Job Array: Groups of small bricks, array tasks, whose size matches the requested cores per task. These can be placed anywhere on the CPU nodes. Job tasks are independent, so do not need to be on the same node and not all pieces must be placed at once. Any remaining pieces are placed on the card and left in the Pending Queue.
- MapReduce: Consist of two steps: a Map step and Reduce step. The job cards are held together with a paper clip. The Map step is scheduled like a Job Array. The Reduce step remains in the Pending Queue until the all tasks of the Map step have completed (job dependency).
- Shared/Large Memory: Represented by an individual brick whose size matches the number of cores requested. As each job is made up of a single piece, it cannot be placed on more than one node block. In this way it models shared or large memory applications, which cannot run across multiple nodes.
- MPI/Distributed Memory: Groups of 1×1 bricks that have been tied together with a string, representing the network. Individual tasks can be placed on different nodes, as they are connected and can communicate over the network (distributed memory).
- GPU: GPU jobs are represented by 1×16 bricks to occupy half of the GPU node. Each GPU node has two GPUs and can run two GPU jobs at a time.

The turn is over when there are no more pending jobs that can start. The students continue to take turns until all the jobs have started.

Where in-person game play is not possible, the "Be the Scheduler" game can be played virtually over a video web-conference call. The facilitator has the game setup on their desk with a camera showing the board and cards in play. The game proceeds as it would in person, where players take turns identifying completed jobs and launching new jobs, with the facilitator rolling the due, drawing the cards, and removing and placing the physical pieces.

B. Discussion

We have run this game twice in person and once virtually during an annual Practical High Performance Computing (PHPC) workshop. Before the game the instructors demonstrated a few rounds to show how the game is played. In person, players were split into two groups of about six, so no one had to wait too long for their turn, with one instructor for each group to assist in game play and answer questions. Each group got their own board, set of cards, job pieces, and die. After the game the two groups reconvened for a group discussion to debrief with discussion questions. The virtual game was played in a single group.

While playing the game the students were able to see first hand why a given job didn't start, and how the scheduler handles different job types. In particular, they noticed that Shared Memory, Distributed, and the Reduce steps of MapReduce jobs sometimes waited several turns in the Pending Queue before starting. They were able to see the difference between a Reduce step sitting in the Pending Queue because it had a dependency on a Map step versus a Shared Memory job stuck waiting for resources. Students also experienced the case where, even though the Distributed job was at the front of the Pending Queue, tasks from the Job Array behind it started because they could start independently, whereas tasks from the Distributed job had to start together. The lesson here is not only why a particular job didn't start, but that they should put in the effort to find out exactly what their job requires and request only that, rather than requesting more than they need.

III. Example 2: Be the Processor

As HPC architectures evolve, shared and distributed memory computing are joined by hybrid computing paradigms that match hierarchical hardware design. To help students develop intuition about creating applications for modern HPC systems, the authors created a game where the participants collaborate to perform the actions of an HPC system to complete a task.

In this game, students take on the role of compute cores, performing tasks as part of a node within a larger cluster. The goal is to work together to create a phrase or sentence from individual data elements. The only requirement on the phrase used for the game is that it must have enough characters to provide each student with roughly three instructions to complete. The learning objectives, achieved through hands-on activities, include understanding the following concepts:

- shared versus distributed memory,
- communication and communication latency,
- load balancing,
- memory spaces, and
- data distributions.

A. Game Play

The student activity mimics the work of a cluster of multi-core nodes. Students are distributed into teams where the team represents a node and members represent processors on the node. The task for each processor is to take a number (input data) and convert it to a character using a cipher, and write or send it to the appropriate rank in the class "world". Each team receives the cipher to decrypt the input data, a card with their node number within the class "world", a representation of local memory space marked with individual memory slots, and a stack of sticky notes to write the characters into memory. In addition, each participant was given instructions, a stack of index cards, and a number representing their MPI rank (global). Each index card had the data required for the task and the memory location where the output (character) was to be stored. Note that the MPI rank is assigned so that students are working locally within the node, shared memory, but understand their location within the larger distributed system.

Play begins once the participants are assigned to their nodes and receive their packet of instructions and data. Within the local node teams, each player completes their tasks individually and writes the result to local (shared) memory, as shown for the in-person game in Figure 3. Once this first step is complete, each team, or node, holds a phrase on their memory cards. In the second step each team sends their portion of the quote to

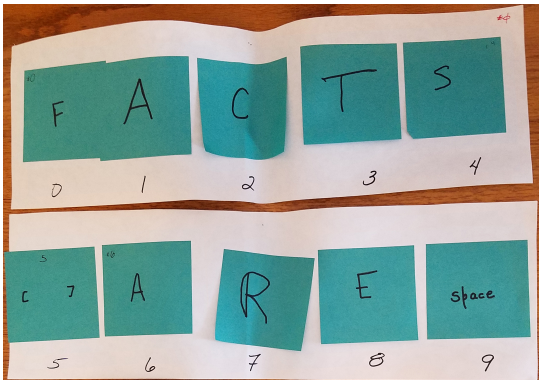

Fig. 3: Memory register for in-person game.

global Rank 0. When Rank 0 receives all of the messages, the full message is posted. Alternatively, node leaders, indicated by the lowest rank (number) within the team, post their completed message to a "buffer" location on the classroom wall that corresponds to their node number. This modification results in physical movement around the room providing a clearer representation of data movement between nodes and allows learners to see communication latency, asynchronous messaging, and messages arriving out of order.

B. Discussion

The in-person game was played with computer science and engineering professionals who expected to use supercomputing resources for data science applications. In 2021, a virtual version was played with graduate students from a range of disciplines. The discussion directly after the game suggested that the game achieved many of the learning goals described earlier in this section.

During the post game discussion, the primary take-away was a more complete understanding of the requirements for message passing in parallel and distributed computing as well as some of the challenges associated with message passing. For example, there were a few cores who were noticeably slower causing the entire team to wait for data before posting their portion of the message. This led to a short discussion about asynchronous behavior and load balancing.

IV. Beyond Our Classroom

The value of these games is broader than HPC. They are designed for those who use HPC systems, across domains in engineering, science, and more. The Processor Game provides an entry to discussions of communication latency and data distributions among non-computing STEM majors, while the Scheduler Game introduces the role of a resource manager.

Further, the lessons that we learned are applicable to other educational games outside HPC more generally. First, it is important to play a round of the game or demonstrate the task so that students understand what is expected. As mentioned by [12] this lowers the cognitive load. Also be aware that players may implement run-time modifications that you hadn't considered, especially for collaborative games. These changes may not necessarily hinder the purpose of the game, so it is important to be flexible. For example, when playing the Scheduler Game one team of students elected a single person to place jobs on the board and the rest decided together where they should be placed. Next, we have found that using tangible, physical components help get and keep the students engaged and involved, giving them an opportunity to work with the material in a different way.

Additionally, games must both match the student background and be relevant to the audience. The placement of the game in the course or workshop schedule is very important. For example, the Processor Game is better suited to a computer engineering course where students need to understand computer architectures and the impact of those architectures on the software applications, or with a student cohort who are engaged in research requiring complex parallel and distributed algorithms. Based on our experience, students need a slightly deeper understanding of HPC architectures and workflows in order to appreciate the challenges and complexities of the game. In the future we plan to play the Processor Game later in the course when students have a better understanding of the motivation behind the game. Additionally, a version that mimics a simple distributed computing paradigm might be a better introduction. The Scheduler Game would be a valuable exercise for anyone who is using or is preparing to use a distributed system with a resource manager. Before playing the Scheduler Game, students should be introduced to the different job types, the components of an HPC system, and the scheduler.

The games are self contained and can be integrated into a curriculum as unique modules. Each of the games takes roughly 30-45 minutes for game play and discussion, assuming students have the background described above. The games can be modified to support slightly different or more specific learning goals, for example changing the distribution in "Be the Processor" would lead to a different set of questions, observations, and discussions. Additionally, modifying the pseudo-code provides an opportunity to explore various HPC workflows. The Lego brick format of the "Be the Scheduler Game" allows instructors at other centers to customize the "cluster" to more closely resemble their hardware and common job types. To assist other workshops and courses, the authors have created Instructor Guides for each of the games and will be linking them to the SIGHPC Education Chapter Outreach Repository [13].

V. Conclusion

While playing the games student were engaged and actively working to understand the concepts around which the games were designed. They discussed their observations with their team members and asked key questions. Overall the experiential learning and engagement resulting from the use of serious games in an HPC education context helped the students develop greater intuition about HPC. Our next steps focus on understanding and developing the appropriate evaluation metrics to confirm the efficacy of the games.

References

[1] SIGHPC Education Chapter, 2019 (accessed June 15, 2021). [Online]. Available: https://sighpceducation.acm.org/resources.html

[2] Shodor and XSEDE, "HPC University," 2020 (accessed October 28, 2020). [Online]. Available: http://hpcuniversity.org/

[3] XSEDE, "User Portal–Online Training," 2019 (accessed October 28, 2020). [Online]. Available: https://portal.xsede.org/online-training

[4] "Archer–online training," 2019 (accessed October 28, 2020). [Online]. Available: http://www.archer.ac.uk/training/online/

[5] K. Kapp, *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Wiley, 2012.

[6] L.-m. Putz and H. Treiblmaier, "Increasing knowledge retention through gamified workshops: Findings from a longitudinal study and identification of moderating variables," in *Proceedings of the 52nd Hawaii International Conference on System Sciences (2019)*, 2019.

[7] C. C. Abt, *Serious Games*. Viking Press, 1970.

[8] M. Ulicsak and M. Wright, "Games in education: Serious games," June 2010 (accessed October 28, 2020). [Online]. Available: https://www.nfer.ac.uk/games-in-education-serious-games

[9] G. Petri and C. G. von Wangenheim, "How games for computing education are evaluated? a systemic literature review," *Computers & Education*, vol. 107, pp. 68–90, 2017.

[10] Computer Science Education Research Group at the University of Canterbury, New Zealand., (accessed June 24, 2021). [Online]. Available: https://csunplugged.org/en/

[11] R. K. Raj, C. J. Romanowski, J. Impagliazzo, S. G. Aly, B. A. Becker, J. Chen, S. Ghafoor, N. Giacaman, S. I. Gordon, C. Izu, S. Rahimi, M. P. Robson, and N. Thota, "High performance computing education: Current challenges and future directions," in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 51–74. [Online]. Available: https://doi.org/10.1145/3437800.3439203

[12] A. DeGloria, F. Bellotti, R. Berta, and E. Lavagnino, "Serious games for education and training," *International Journal of Serious Games*, vol. 1, January 2014.

[13] SIGHPC Education Chapter, "Outreach repository," 2019 (accessed October 28, 2020). [Online]. Available: https://github.com/SIGHPC-Education-Chapter