HPC Tunathon Attendee Guide
Version 0.1

Brian J. N. Wylie
Jülich Supercomputing Centre

Notice: *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 676553.*

Contents

1 Introduction

An HPC *Tunathon* is a **multi-**day coding event in which teams of developers tune and optimize their application(s) to execute and scale on one or more HPC computer systems. If you don't already have your code running (correctly) across multiple compute nodes, then the workshop is not for you. Teams should consist of **one** or more developers who are intimately familiar with (some part of) their application, and will be paired with one or more expert mentors. The **mentors** come from the hosting HPC centre (system and application support), hardware/software vendors, and possibly elsewhere.

2 Preparation

Recommendations have been gathered from participants and mentors of previous (similar) events over many years. While each team will proceed differently in practice, working towards its own goals at its own pace, teams that follow these suggestions can be expected to have the greatest success.

2.1 Team introductions

Leading up to the event, the local organizer will send an email introducing your team to the mentors who will work with you. Your team should then schedule a tele/videoconference to address the following:

- Introduce Team/Mentor
- Discuss code(s) your team will be working on (providing code access where appropriate)
- Determine which compute system(s) and toolchains your team will use during the event
- Assign any action items based on discussions

This initial meeting will help get your code team organized and ensure everyone is on the same page for follow-up discussions.

3 Access to the Tunathon compute system(s)

It is highly beneficial for all team members to obtain access to the compute system(s) they intend to (or might possibly) use at the Tunathon *well before the start of the event*. This provides an opportunity to familiarise with aspects of the system that might be new to you (e.g., a different batch scheduler, job launcher, etc.) and also get your application compiled and running (correctly) on the system(s). This helps ensure your team is ready to start using the system(s) productively when you arrive for the event, instead of spending time learning how to use the system(s).

Should you need, or wish, to use compute nodes with additional resources (e.g., memory or GPGPUs), make sure this is agreed with the local Tunathon organizer in advance.

NB: It might not always be possible to obtain access to the compute system(s) before the event due to institutional restrictions, however, you should always ask and be prepared accordingly.

4 Preparing code to run on Tunathon compute system(s)

Although you might be using your application for production work already, that does not ensure that it is ready for the work you will be doing at the Tunathon. For example, if your code takes an hour to compile and four hours to produce results, it will be difficult to test the many incremental changes you will likely be making to your code during the event. This section outlines recommendations for preparing your application for the event.

Your code should be self-contained whenever possible, and external dependencies (e.g., netCDF) need to be explicitly identified in advance. Make sure that the local Tunathon organizer is informed of the libraries (and any other tools) that you require to be installed, including precise version and configuration details. Alternatively, or if you require unreleased versions or non-standard configurations, you can build these yourself. Limiting external and system-specific dependencies will make it easier to get your code running on Tunathon computer systems.

After making any changes to your code (e.g., using a different compiler or compilation/optimization options, extracting kernels, building your own libraries), you should always confirm that your application still compiles, runs, and (of course) gives acceptable results. Ideally, this should be done on the computer system(s) you will be using during the event. This workflow (modify – compile – run – verify) will used frequently during the week of the event, so it is important that the process is efficient.

In general, you should prepare at least two distinct configurations of your application. A small/short version which runs on a single compute node in several minutes for initial/quick testing and verification, and a realisticly-large version capable of running (efficiently) on several compute nodes. Depending on whether you expect to pursue strong or weak scaling, you may need different suitably-sized datasets.

5 Understanding your code and simplifying your workflow

It is likely that the mentors working with your team are not familiar with your application (at least not in detail). It is therefore helpful to have a way of describing your application's program flow to them (e.g., via a call tree or flow chart). Helping your mentors (and all team members) understand your code structure before the event can make them more efficient and save a lot of precious time during the event.

Having a profile of your application, which shows details about how much time is being spent in different regions/routines of the code, facilitates identification of the most beneficial regions to focus on for optimization and tuning. For example, you want to ensure that you work on parts of your code that account for a sizable percentage of the total runtime (of the realistic testcase); optimizing regions/routines that only account for 1% of the total runtime will not gain you much. Help will be provided before the workshop to generate a high-level profile, suitable for initial discussion and planning, with more comprehensive performance analysis during the Tunathon itself.

Another important aspect of preparation that is often overlooked is having a way to verify the correctness of your results. This is an important part of the Tunathon, so arriving with an automatic means of doing so (e.g., comparing output against a validated reference) can save more time for development. It is not uncommon for optimized code to run blazingly fast but

not give correct results!

6 Attending a Tunathon

During the Tunathon, your team will work alongside your mentors on the goals you have (hopefully) set and agreed during the pre-event meetings. In addition, there are presentations and morning updates that your team will be required to participate in.

6.1 Introduction presentations

On Monday morning, the first order of business will be for one member of each team to give a short (5–7 minutes / 5–7 slides) presentation introducing their team. Ideally, this should include an introduction of your team members, a brief description of the application, an outline of its structure (highlighting the most expensive parts), and your goals for the week (e.g., the parts of your code you will be targetting, anticipated scaling, etc.) Remember that most participants will probably not be familiar with your specific domain, so adjust the level accordingly.

6.2 Morning updates

On Tuesday through Thursday mornings, each team will give a short (3–4 minute / 2–3 slides) update, including

- Progress make since last update
- Goals for the day
- Problems currently faced
- Problems already resolved (that other teams might find useful)

In addition to sharing your progress, these updates are good opportunities to get feedback from other teams and mentors. There is a chance that a problem you are facing has already been encountered and resolved by someone else. Or you might have found and resolved a problem (or reported a bug) that other teams might currently be facing (or about to run into). Tunathons are meant to be cooperative events among all participants, and these update sessions are central to that theme.

6.3 Final presentations

On Friday morning the teams will finish up their development work for the week, and after lunch give a final presentation (5–7 minutes / 5–7 slides) detailing their accomplishments; issues they ran into, how they resolved them; speed-ups and scalability improvements obtained, as well as their closing thoughts on the event and what they learned. The final day typically formally ends around 15:00.

7 Feedback and suggestions

We are always looking for ways to improve these events. If you have feedback you believe would help improve this document or the Tunathons themselves, please send us your suggestions.