Programming Project: Hybrid Programming

Rebecca Hartman-Baker

Oak Ridge National Laboratory

hartmanbakrj@ornl.gov

Hybrid Programming Project

- Laplace Equation
- The Code
- Its Performance
- Your Project

Laplace Equation

- Second-order elliptic partial differential equation:

$$\nabla^2 u = 0$$

- Simplest of elliptic PDEs, applied in many application areas (e.g., electromagnetism, astronomy, fluid dynamics)

Laplace Equation

- We will solve 2-D Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

with boundary conditions

$$u(x,0) = \sin(\pi x), \quad 0 \le x \le 1,$$

$$u(x,1) = \sin(\pi x)\mathrm{e}^{-x} \quad 0 \le x \le 1,$$

$$u(0,y) = u(1,y) = 0 \quad 0 \le y \le 1.$$

and analytical solution

$$u(x,y) = \sin(\pi x)\mathrm{e}^{-\pi y}$$

The Code

- I downloaded “interesting” MPI implementation of Laplace equation from the internet
  - You can find anything on the internet!
  - Actual URL withheld to protect the innocent ☺
- This program is great example of both things to do and things not to do
  - Fixing codes like this and scaling them to full machine (224,256 cores on Jaguar) is part of my job description
  - If you get a kick out of doing this project, let me know when you are graduating! ☺

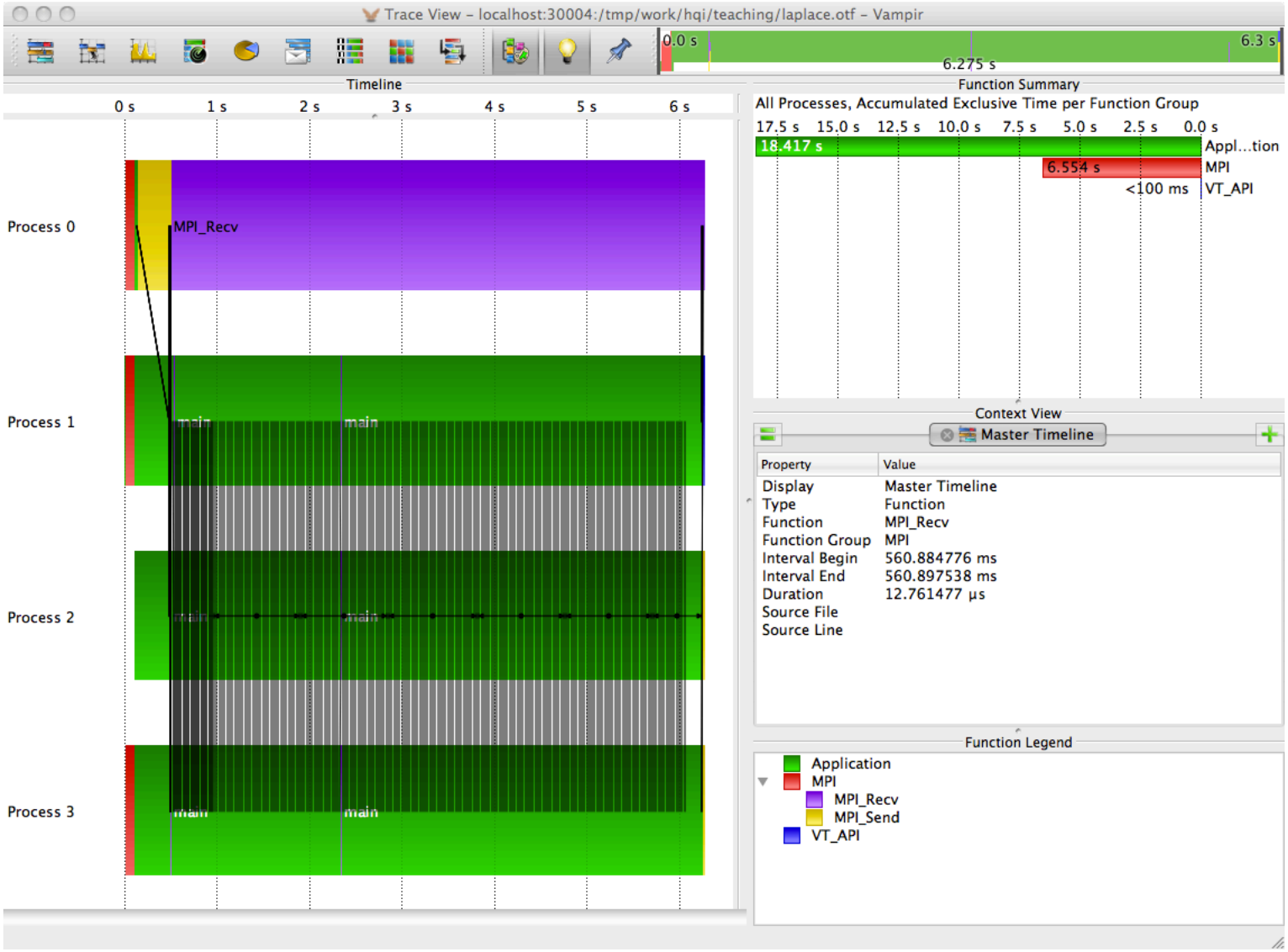
The Code

- Pure MPI program
- One process is manager, remaining processes are workers
- Manager just manages (does not perform computations)
  - Bad idea? Yes and no. (but “yes” as this code is implemented)
- Workers receive info from manager and perform calculations
- After they finish, workers send results to manager who prints them to output file
- Comments in code indicate that student also wrote a hybrid implementation
- Anyone curious I will provide with hybrid implementation, but probably best not to be influenced by it ☺

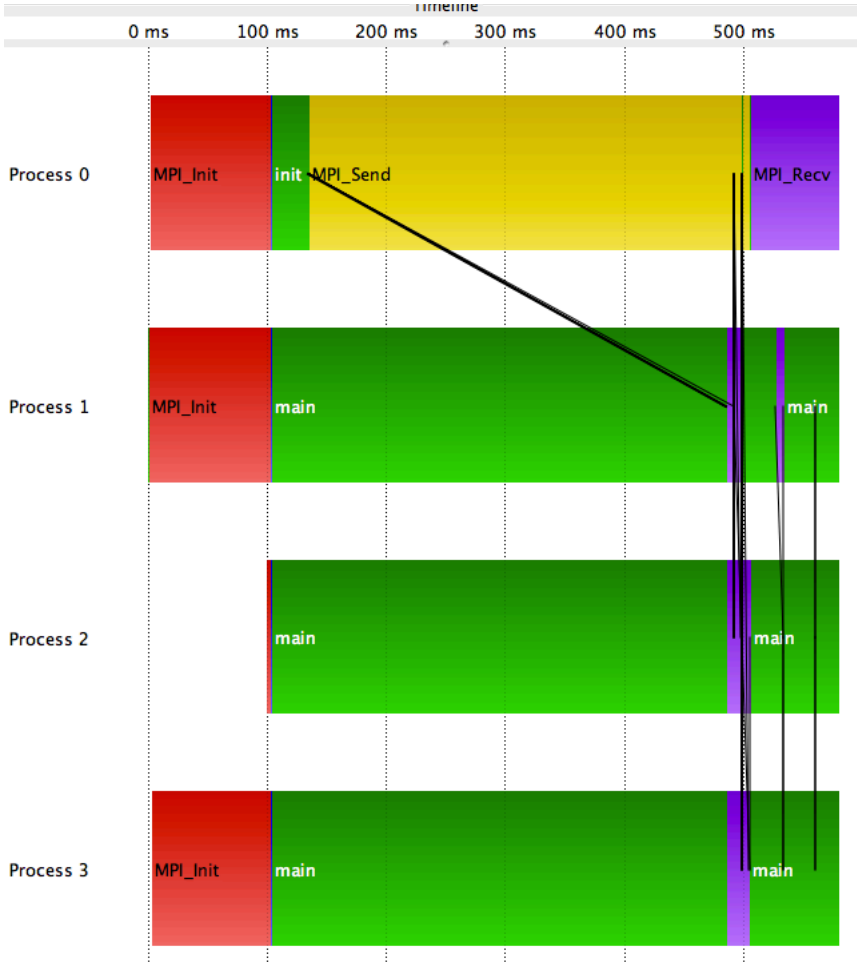
Performance

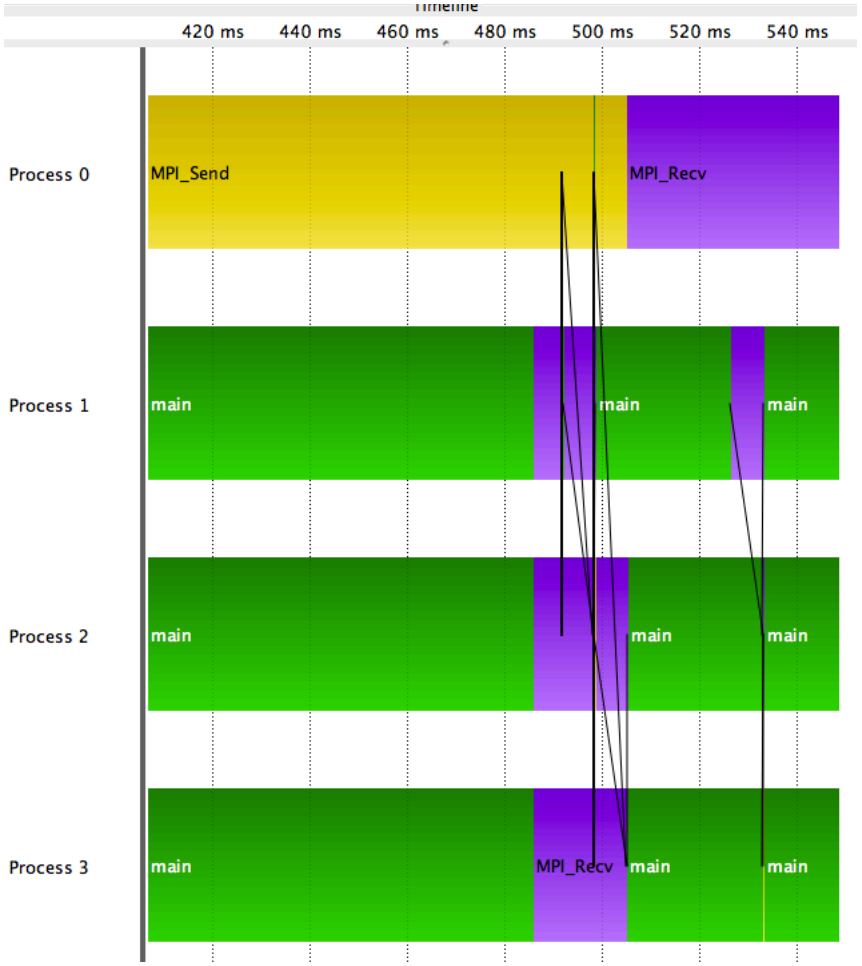
- Ran on ORNL's Jaguar, 4 processors, roughly 6 seconds
- Recompiled with VampirTrace instrumentation, and viewed trace with Vampir (my new favorite tool)
- Very interesting trace results

Performance: Overall Trace

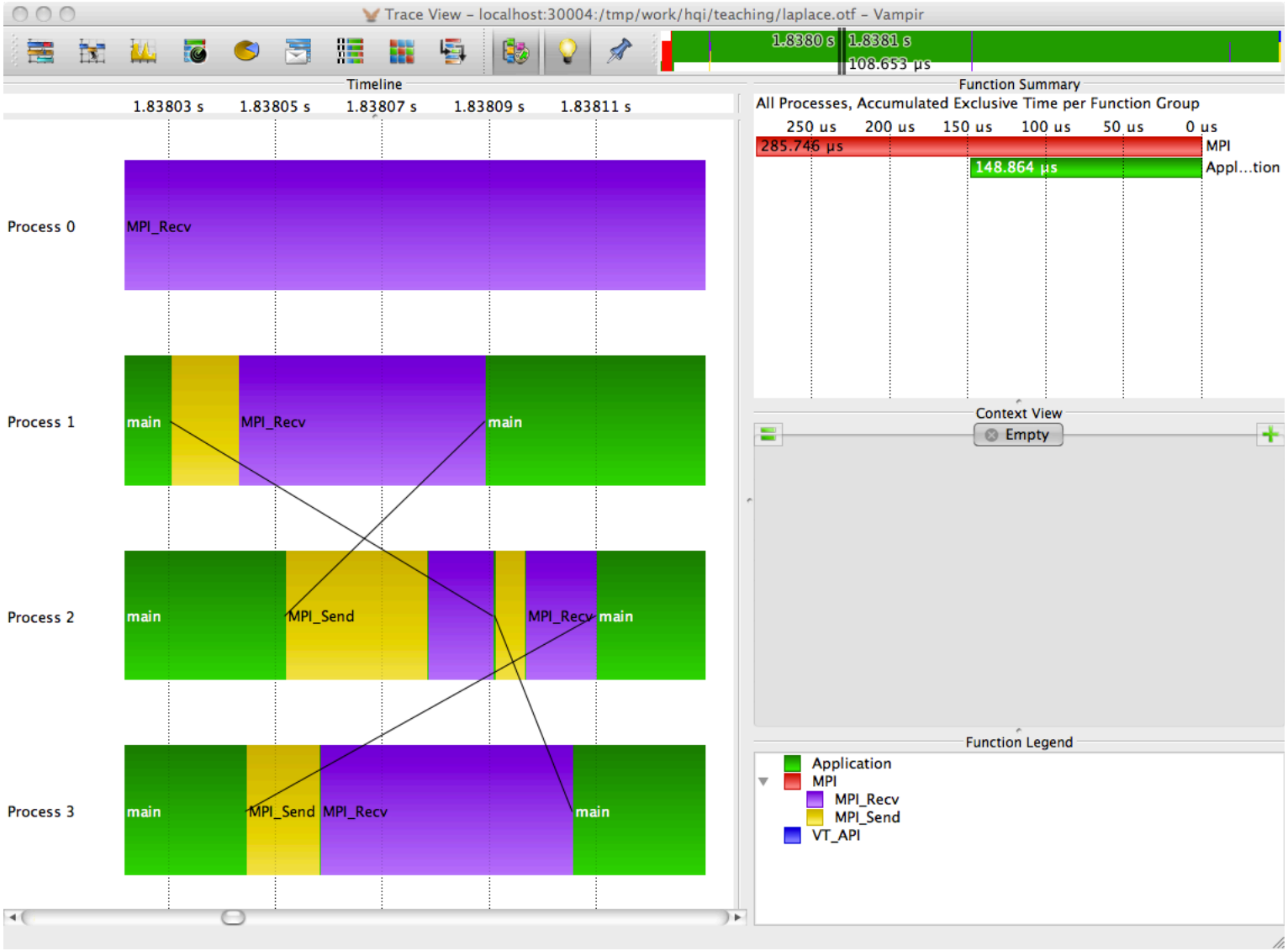


Performance: Initialization

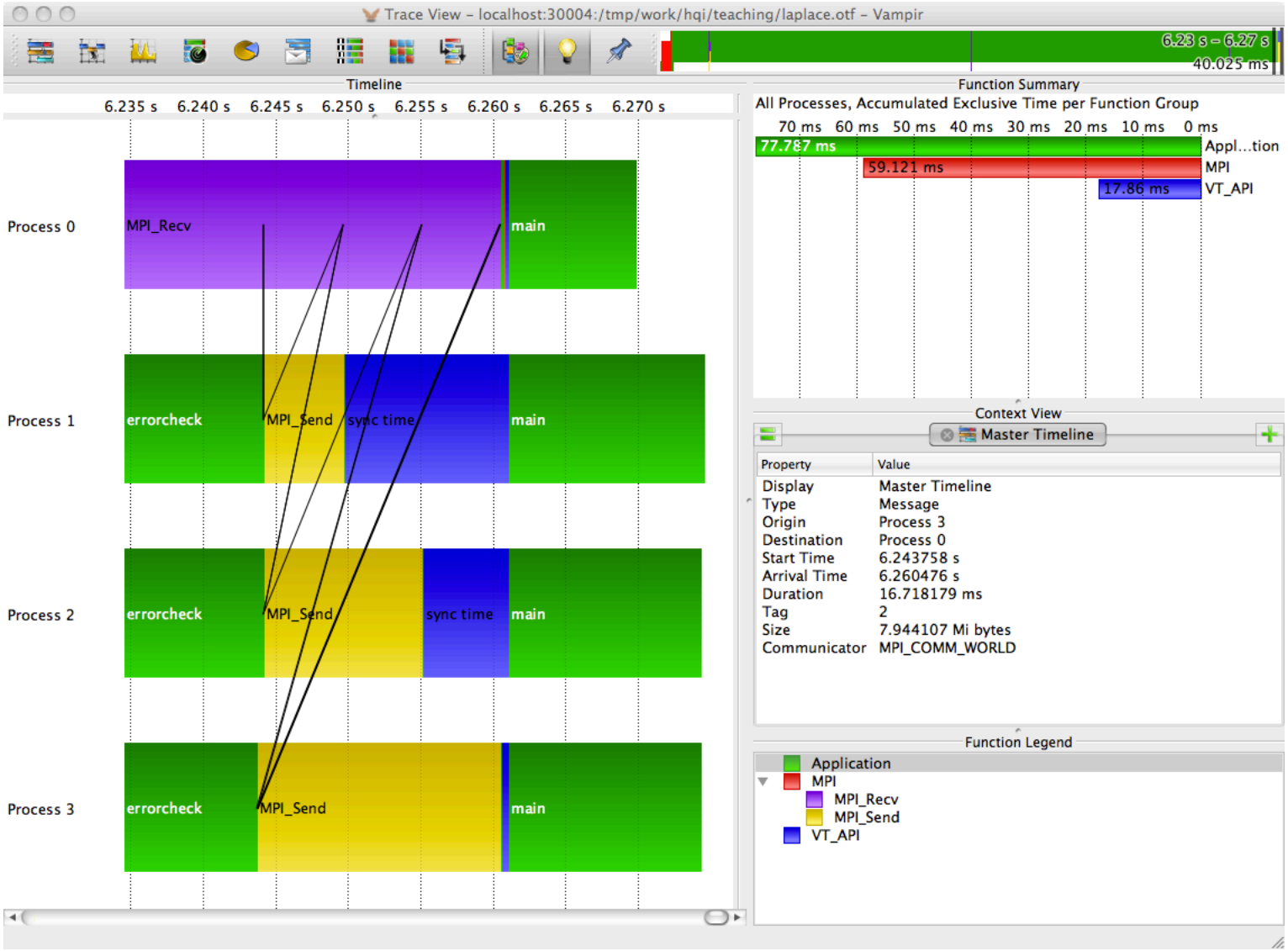




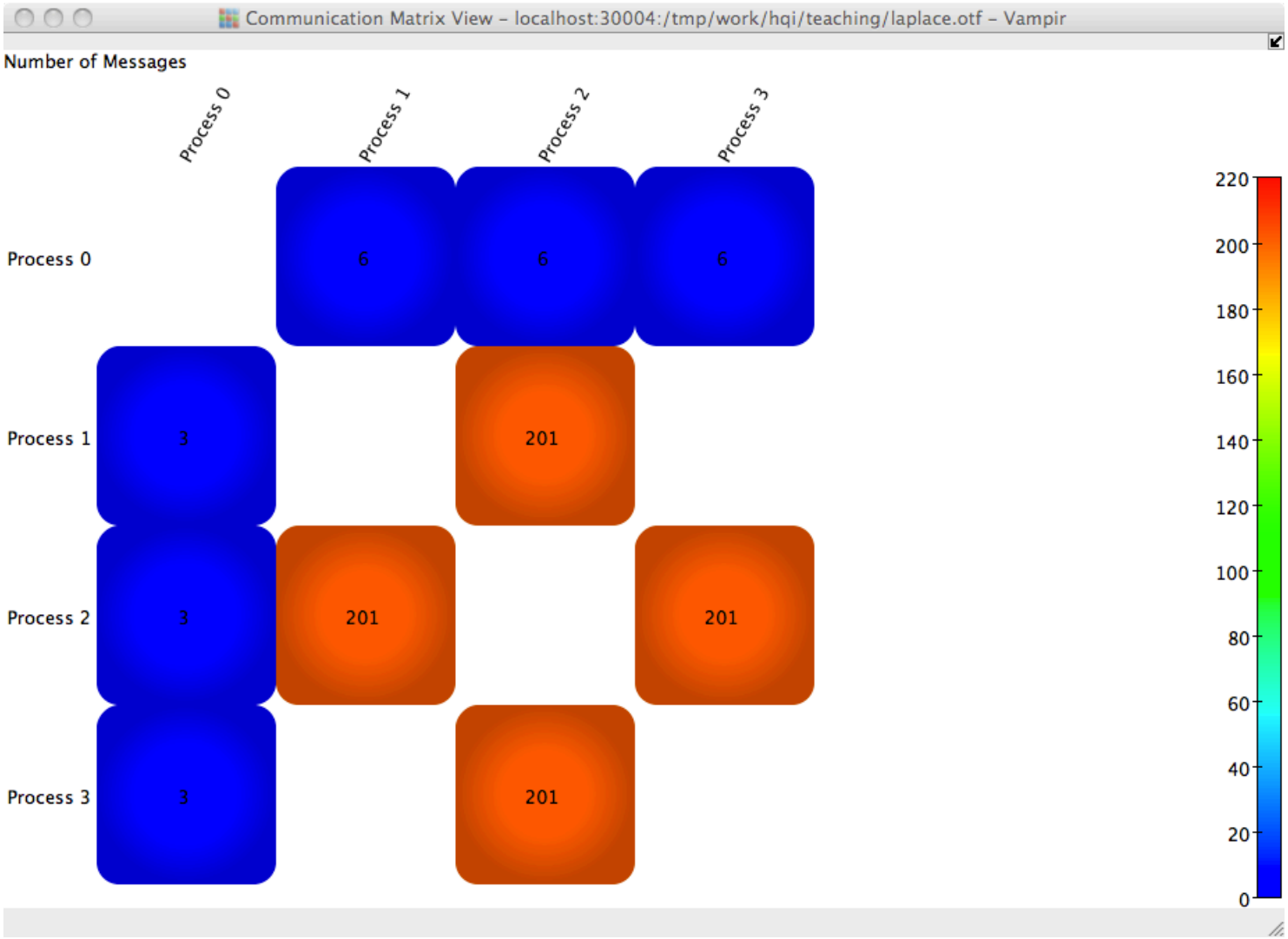
Performance: Load Imbalance



Performance: Finalization



Performance: Communication Matrix



Your Assignment

- Fix this code!
  - First, test for correctness (code runs, but does it work?) and fix, if necessary
    - Trust me, this is important first step any time you work on ~~someone else's~~ code!
  - Add OpenMP directives
  - Add threading support in MPI
  - Rewrite manager-worker paradigm so manager does some work too
- Things to consider
  - Message-passing: who sends to whom? What performance gains (if any) can be attained by threads sending messages? (Hint: try it and see!)
  - Is there a case for having a manager process (either MPI or OpenMP) that only communicates?