Due date: Wednesday night

Motive: need time to myself, and then goal of coming back.

can’t be talked about negatively, and but also need to stay away

General goal: inform

Specific Goal: you are important to me, even though I also need time to myself.

Ulterior Motive: I want to stay connected, for a job later.

…

Point: Tell you about the conversation after the defense.

This email is a continuation of our conversation from after the defense. I aim to expand on certain points from the conversation.

Point: software was important to me, (though I don’t want to write more)

First, I feel that the work done at LLNL on software development was important for the thesis, and should be discussed more. The software provides the infrastructure for and core of the technique of my thesis (not explicitly stated in the thesis). The software is about 65% of all the code written for the thesis. More broadly, while the thesis talked about the application programmer’s effort, it didn’t talk about the developer’s effort. Each of the three optimizations of Chapter 5 required at most 4 lines of code to be added to the software and library implementation of static/dynamic scheduling, for a total of 10 lines of additional code needed for all of Chapter 5’s optimizations. In this light, the implementation of the technique is as important as the technique itself, since the implementation dictates how easy it is to experiment with further techniques. In short, I wanted to bring up the software development done at LLNL further because of its importance to my work, and also to the development of my skills.

TODO: development of skills

TODO: bring up 🡪 bring further

TODO: developer’s programming effort,

TODO: ending of paragraph

TODO: not explicitly stated in the thesis – parens

TODO: In this light,

Todo: Maximum among three optimizations

TODO: software vs library implementation

TODO: Experiment

TODO: Discussed more

TODO: Because the

TODO: Defines

TODO: understanding

Point: important to show happiness, but I want to stay away.

Second, you said something about happiness. I wasn’t the best scholar I could be, and it was because of my happiness. First and foremost, I want to apologize that I wasn't the best I could be. With that said, I don't think I could be the happiest by continuing to do work in a position similar to the LSP position, or a position similar to the one I’m doing now, at least for the time being. Whatever the justifications or combination of justifications for it, a desire for increased happiness captures a large part of the reasoning that I intend to find a job outside of HPC, but that makes use of HPC knowledge. The thing to point out though is that my experience at LLNL provided me with the grounds to think about happiness in the combination of technical work - having stayed at UIUC would not have done this.

TODO:

TODO:

Point: forgetting about that conversation, impacted me

There’s one last thing, and this will be as close to personal life as I intend to get in this setting. We brought up a meeting/conversation from a couple years ago from after the SC12 conference in Salt Lake City. In that earlier conversation, I talked about parts of my social life. We then talked about depression. It was necessary at the time to have such a conversation, and it was good to talk about it once more in the context of when we met a few months ago, but I plan to forget about it now. The thing that comes out of this conversation though is to ensure such problems do not interfere with work by taking measures such as talking to people about problems regularly, and more strongly (as you mentioned when we spoke a few months ago) to talk to a professional, e.g., psychologist, psychiatrist, regularly. I have taken such measures in the last several months, and plan to take more measures in the coming months.

TODO: this earlier conversation

TODO: Depression

TODO: from a couple years ago

TODO: said things 🡪 talked about

TODO: measures 🡪 <omit>

TODO: once more

Point: important to me, but don’t want to get too close.

I wrote the above to tell you that the experience at LLNL had some meaning to me, in the context of the conversation we had. We hadn’t talked all of what’s above when we spoke a few months ago. I got time to say something more last night and today. I felt that I should talk more about it, e.g., for reference letters, and also to talk about things in a more open manner.

TODO: a few months ago

TODO: figure out late last night and this morning

TODO: simply

TODO: things

not simply for reference letters , but also to talk about things in a more open manner.

to work with you in the future, if given the opportunity.

this,,

1. Includes people I interacted with – 15 mins , done
2. I aim to highlight points in our conversation – 5 mins. – done
3. Emphasized in our conversation - 10 mins -done
4. Knowing happiness
5. Taken in the last several month – 5 mins
6. This does not interfere with work
7. principles learnt from the code since I left, and had – 6 mins
8. add two sentences at the end – 5 mins
9. 🡪 ~~happiness the reason for not achieving a path to those visions was due to my technical abilities~~ 5 mins Action: take out
10. ~~distinguish between happiness and satisfaction~~
11. ~~make appropriate point about what LLNL did in terms of happiness.~~
12. As my supervisor, - 10mins
13. I ~~wanted to say something more~~ – 5 mins
14. ~~build upon and finish the work for the thesis. –~~ 5mins
15. ~~[, as things have just started to cool down a bit]. Action : take out because redundant – 1 mins~~
16. ~~these things~~
17. ~~this~~
18. ~~Iwasn’t the best scholar I could be - done~~
19. ~~Should be discussed more.~~
20. Provides infrastructure for core of technique
21. Provide details in second paragraph
22. Explain point about organization
    1. Option 1: Less code for new strategies.
    2. Option 2: Less code for application programmers
    3. Option 3: More efficient code because efficiencies handled and less error-prone
    4. Option 4: Organization is most important in HPC. innovation
23. I felt it should be discussed more
    1. I felt it should be brought up more because of its importance to my work.
    2. “” important role to my work.
    3. And
24. Substantiation of Less programming effort
    1. Large change between straightline version and software version
    2. Less change from one version to the next
25. ~~More broadly,~~
26. Near-duplicates
27. Transition between core of implementation and paragraph
28. If you don’t count the intermediate scheduling strategies composed/created for illustrative purposes to show in Ch. 6 (never used) composed scheduling strategy for illustrative purposes
29. ~~went down by at least half the original implementation that I wrote when I started the work 🡪 Further substantiation of less programming effort - explain original and new code.~~
30. felt it should be discussed more because of its importance to my work, and also to my skill set.
31. Perhaps the biggest software development tactic is to find commonalities in different optimizations, as opposed to hacking a new optimization for each.
    1. Perhaps the biggest software development tactic is to find commonalities in different optimizations, as
    2. Perhaps the biggest software development tactic is to find different optimizations and.
32. ~~5 lines added over the original implementation of static/dynamic scheduling and 10.~~
33. ~~In this light, the implementation of the technique (or basic technique which is extended upon, in Chapter 5) is almost as important as the technique itself.~~
    1. In some ways, this should work hand in hand with the techniques themselves.
    2. Spaghetti code
34. ~~Debugging :~~ 
    1. Debugging didn’t take time
    2. An important issue is to solve problems through
35. ~~Technique allows for innovation~~
36. ~~We talked briefly about it, but should expand more since it wasn’t in the thesis.~~
37. Try further techniques
38. The maximum among the three individual optimizations was 4 lines over the software framework and basic implementation of static/dynamic scheduling to do the optimizations in Chapter 5, for a total of 10 lines of additional code needed.
    1. Among the three optimizations of Chapter 5, a maximum of 4 lines of code were added to the software and library implementation of static/dynamic scheduling, with a total of 10 lines of code added for the optimizations.
    2. Among the three optimizations of Chapter 5, the maximum lines of code changed to the software and library implementation of static/dynamic scheduling were 4, for a total of 10 lines of additional code needed for the optimizations.
    3. Among the three optimizations of Chapter 5, a maximum of 4 lines of code were added to the software and library implementation of static/dynamic scheduling, for a total of 10 (4, 3, 3) lines of code added for the optimizations of chapter 5.
39. find a job outside of HPC, and that makes use of HPC knowledge – light
    1. take a break for some time off , figure out a plan for work, and take time off.
    2. Find a job in a small company, and that preserves and makes use of HPC knowledge.
40. Yet, -don’t need this, since we want to
41. ~~The thing to point out though is - good to point out transitions wise.~~

I also think that there is a level of greatness which I think you are going after. It is the ability to formulate an idea about computing in plain English. I am writing somewhat freely without paying attention to writing.

some technical concepts.

I think this is admirable – and I would like to be part of this effort.

This code is used within our software infrastructure daily. It forms a key part of code for my thesis.

However, I plan to not lose my presence in the area of HPC. I will still stay involved in publication work, say 25%.

I wanted to write this because I want to tell you that I value your input and feedback

to directly acknowledge you and Todd.

I felt that this had to be emphasized from our conversation, because

And I didn’t get a chance to acknowledge you and Todd (it’s in several of the slides and the thesis acknowledgements).

because the code developed at LLNL, specifically the software infrastructure, plays an important role for the work I did with Prof. Gropp and for the thesis.  
  
because you are my supervisor, and I simply am unsure that I have been talkative about this.

to build upon and finish the work for the thesis.

. Distinguish between happiness and satisfaction. By working at LLNL for the Lawrence Scholar, I gained knowledge of what makes me happy.

My reasoning for it is technical ability. But, it may also be circumstancial.

I want to mainly explain that while I left, I still continue to think about LLNL, and that this experience has significance on my resume. When we spoke, I got a different impression, and I just want to basically say that no

More broadly, I had continued to use the skills acquired through developing the software while at UIUC. For example, the code is written in C.

A distinguishment must be made between happiness about work and happiness in life. Also, a distinguishment must be made between happiness about work and satisfaction (as in job satisfaction).

More broadly, the systematic software development and implementation that I learned at LLNL were followed for the macro-based scheduling (vSched) developed even after I came back to UIUC.

You can also make theories. This is private stuff. The line is drawn there.

I've mentioned several factors, but the happiness part, at a high-level captures the reason that I wouldn't be happy to continue the work.

In fact, it makes me unhappy seeing myself trying to realize those visions.

I had visions earlier before starting the LSP (the LSP application’s 1-pager gives an idea of those visions), but I don't think I’ve been able realize those visions, for both technical and non-technical reasons.

. Perhaps one thing that came close to describing being the happiest is exactly the 1-page LSP document in the LSP application, which answers the question of where I see myself in the 10-year time frame. I couldn’t achieve a path to these visions. In my mind, the reason for not achieving a path to those visions was due to my technical abilities. It could also be circumstantial.

Perhaps one thing that came close to describing being the happiest is exactly the 1-page LSP document in the LSP application, which answers the question of where I see myself in the 10-year time frame. Whatever the case, the

Compared to other places, LLNL is the only place I’ve found this.

Broadly speaking, I had continued to use the software developed at LLNL while at UIUC, and plan to continue to use these skills and experience.

This includes test cases and error-handling. Often times, innovation is considered to be done in a random environment where freedom is necessary.

An additional point is that the software is beneficial for other programming models, which may be used in future exascale systems. A particularly interesting one is MPI+MPI-shm. An additional point is that this software forms a basis for other programming models. The software has been used on a daily basis since the week after I came back to Illinois from LLNL.

Doing this reduces the lines of application code changed compared to the computational loops arranged by hand. The deeper thing is that systematic organization actually makes code easier

More broadly, I spoke about the application programmer effort (lines of code needed to apply our strategy to a code), but I didn’t talk about the developer programmer effort (lines of code needed to implement the strategy).

I thought this was important, because even though it was satisfying to write lots of code, it was painful, and I found that creating a library helped reduce the pain

The lines of code needed for implementation went down by at least half the original implementation that I wrote when I started the work (25 to 10). This is an important measure, because it signifies how easy it is for the developer to extend the work. This has significantly reduced the lines of code needed for development of new scheduling strategies, important to developers. I plan to make this code even better when I have a chance in the next few months.

Chapter 6, i.e., for the block diagram, but not

As you go up the diagram, the most lines of code changed was 5, at the very end for the final combination sched. A basic scheme is to -. The lines of code needed for implementation went down by at least half the original implementation that I wrote when I started the work (25 to 10). I haven’t figured out how many lines of code.

I felt the above should be discussed more, not just for my thesis work, but for the skills gained

extensibility to future techniques.

Overall, I felt For these reasons,

[Each optimization could be further combined]. This is important to note because it makes compiling easier. This is important because it doesn’t require much effort to handle errors after the additions were made. There are other ways, but these are the main ways, and this is the main one of the software tactics that sticks out to me. This is important to note, because it signifies the ease of extensibility of the software.

Additionally, the time spent debugging was relatively small for the added code.

Developer effort reduced. Substantiate with comparison to the strategy in the first code. Comparing with straight-line code that I have, the strategy requires only a few lines of code to implement in my library.

Think about CALU.

Relevant because it reduced effort to innovate/extend for the developer.

Acknowledge Todd, but don’t say this explicitly.

I felt that this should be brought up more because of its important role to my work, and because it speaks to the skills.

Macro-based scheduling – significantly reduced

Perhaps the biggest software development tactic to consider is to modularize the most important code optimization in the runtime, and then add further optimizations.

find commonalities in optimizations done to application code, and to put these into a runtime. It’s also to modularize the most important optimization to put it into a runtime.

, i.e., lines of code needed to apply our strategy to an application code,

, i.e., lines of code needed to implement the strategy

lines added over the original implementation of static/dynamic scheduling and 10 lines added when all three optimizations were put together for the optimizations in Chapter 5

Perhaps the biggest software development tactic to consider is to find equivalences in the set of different application code optimizations, and then optimize based on the most important optimization within the runtime, and then to add other optimizations over the most important optimization. Perhaps the biggest software development tactic to consider is to modularize the most important optimization in, and then add further optimizations over the most important one.