# A DESCRIPTIVE TITLE, NOT TOO GENERAL, NOT TOO LONG

*Markus Püschel*

Department of Computer Science
ETH Zurich, Switzerland

- The hard page limit is 7 pages in this style. This excludes references and excludes the short, mandatory part on individual contributions (see end).
- No appendix.
- Do not reduce font size or use other tricks to squeeze.
- The pdf is formatted in the American letter format, so may look a bit strange when printed out (and there is no real reason to do this).

## ABSTRACT

Describe in concise words what you do and the main result. The abstract has to be self-contained and readable for a person in the general area. Write the abstract last and keep it short.

## 1. INTRODUCTION

Do not start the introduction with the abstract or a slightly modified version. What follows is a possible structure of the introduction. This structure can be modified, but the content should be the same. The introduction should definitely end on the first page and leave some space for the second section on the first page.

**Motivation.** The first task is to motivate what you do. You can start general and zoom in one the specific problem you consider. In the process you should have explained to the reader: what you are doing, why you are doing, why it is important (order is usually reversed).

For example, if my result is the fastest DFT implementation ever, one could roughly go as follows. First explain why the DFT is important (used everywhere with a few examples) and why performance matters (large datasets, realtime). Then explain that fast implementations are very hard and expensive to get (memory hierarchy, vector, parallel).

List some related work on fast implementations for your project if available (in particular those that you compare against) and very briefly explain what they did.

**Contribution.** Now you state what you do in this paper. In our example it could be: presenting a DFT implementation for sizes divisible by 4 that is optimized for locality and uses AVX SIMD operations. It is faster for some sizes $> 128$ than prior code. Be brief so you have space for the core part of the paper.

The author thanks Jelena Kovacevic. This paper is a modified version of the template she used in her class.

## 2. BACKGROUND ON THE ALGORITHM/APPLICATION

Give a short summary on the algorithm or application that you then later optimize. Show some of the key parts or equations. Then define the cost measure and show the cost analysis that you later use to create performance plots.

This section should end well before the end of page 2.

## 3. OPTIMIZATION PERFORMED

Now comes the "beef" of the paper, where you explain what you did. Again, organize it in paragraphs with titles. As in every section you start with a very brief overview of the section.

For this course, explain all the optimizations you performed. This mean, you first very briefly explain the baseline implementation, then go through locality and other optimizations, and finally SSE (every project will be slightly different of course). Show or mention relevant analysis or assumptions. A few examples: 1) Profiling may lead you to optimize one part first; 2) bandwidth plus data transfer analysis may show that it is memory bound; 3) it may be too hard to implement the algorithm in full generality: make assumptions and state them (e.g., we assume $n$ is divisible by 4; or, we consider only one type of input image); 4) explain how certain data accesses have poor locality. Generally, any type of analysis adds value to your work.

As important as the final results is to show that you took a structured, organized approach to the optimization and that you explain why you did what you did.

Mention and cite any external resources including library or other code.

Good visuals or even brief code snippets to illustrate what you did are good. Pasting large amounts of code or screen shots to fill the space is not good.

## 4. EXPERIMENTAL RESULTS

Here you evaluate your work using experiments. You start again with a very short summary of the section. The typical structure follows.

**Experimental setup.** Specify the platform (processor, frequency, cache sizes) as well as the compiler, version, and flags used. I strongly recommend that you play with optimization flags and possibly compilers, even though this can only be a minor aspect in the project and report.

Then explain what input you used and what range of sizes. Go large enough so the computations takes a few minutes or ideally even longer.

**Results.** Next divide the experiments into classes, one paragraph for each. In the simplest case you have one plot that has the size on the x-axis and the performance on the y-axis. The plot will contain several lines, one for each relevant code version. Discuss the plot and extract the overall performance gain from baseline to best code. Also state the percentage of peak performance for the best code. Note that the peak may change depending on the situation. For example, if you only do additions it would be 12 Gflop/s on one core with 3 Ghz and SSE and single precision floating point.

Do not put two performance lines into the same plot if the operations count changed significantly (that's apples and oranges). In that case, for example, first perform the optimizations that reduce op count and report the runtime gain in a plot. Then continue to optimize the best version and show performance plots.

**You should**

- Follow to a reasonable extent (i.e., don't stress out about it) the guide to benchmarking presented in class, in particular
- plots should be very readable (do 1 column, not 2 column plots for this class), including the font size. An example is below (of course you can have a different style),
- every plot answers a question, which you pose and extract the answer from the plot in its discussion

Every plot should be referenced and discussed (what does it show, which statements do you extract).

Most important: Be analytical about your results. Explain them as good as possible and in the end try to argue why more performance is not possible. Read the FAQs (How the project is graded) on the website.

## 5. CONCLUSIONS

Here you need to briefly summarize what you did and why this is important. *Do not take the abstract* and put it in the past tense. Remember, now the reader has (hopefully) read the paper, so it is a very different situation from the abstract. Try to highlight important results and say the things you really want to get across. Be brief.

## 6. FURTHER COMMENTS

Here we provide some further tips.

**Further general guidelines.**

- For short papers, to save space, I use paragraph titles instead of subsections, as shown in the introduction.
- It is generally a good idea to break sections into such smaller units for readability and since it helps you to (visually) structure the story.
- The above section titles should be adapted to more precisely reflect what you do.
- Each section should be started with a very short summary of what the reader can expect in this section.
- Do not use subsubsections.
- Make sure you define every acronym you use, no matter how convinced you are the reader knows it.
- Always spell-check before you submit.
- Be picky. When writing a paper you should always strive for high quality. Many people may read it and the quality makes a big difference. In this class, the quality also contributes to the grade.
- Books helping you to write better: [1] and [2].

**Graphics.** Fig. 1 is an example plot that I used in a lecture. Note that the fontsize in the plot should not be any smaller. On the other hand it is also a good rule that the font size in the plot is not larger than the one in the caption (otherwise it looks ugly).

**Up to here you have 7 pages.**

## 7. CONTRIBUTIONS OF TEAM MEMBERS (MANDATORY)

In this mandatory section (which is not included in the 7 pages limit) each team member should very briefly (telegram style is welcome) explain what she/he did for the project. I imagine this section to be maximally one column. Do not put any appendix besides that.

Include only

- What relates to optimizing your chosen algorithm / application. This means writing actual code for optimization or for analysis.
- What you did before the submission of the presentation.

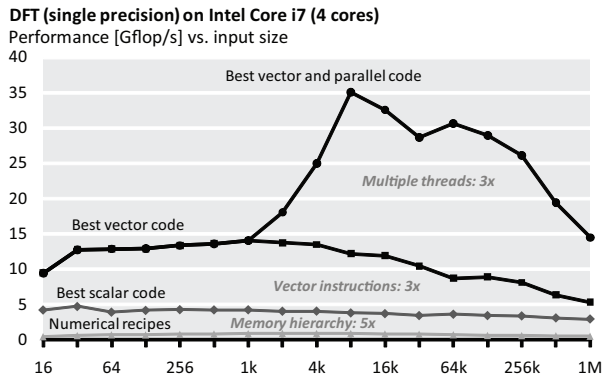


**Fig. 1**. Performance of four single-precision implementations of the discrete Fourier transform. The operations count is roughly the same. *The labels in this plot are about the smallest you should go.*

Do not include

- Work on infrastructure and testing.
- Work done after the presentation took place.

Example and structure follows.

**Marilyn.** Focused on non-SIMD optimization for the variant 2 of the algorithm. Cache optimization, basic block optimizations, small generator for the innermost kernel (Section 3.2). Roofline plot. Worked with Francois on the SIMD optimization of variant 1, in particular implemented the bit-masking trick discussed.

**Hans.** ...

**Francois.** ...

**Isabella.** ...

## 8. REFERENCES

[1] N.J. Higham, *Handbook of Writing for Mathematical Sciences*, SIAM, 1998.

[2] W. Strunk Jr. and E.B. White, *Elements of Style*, Longman, 4th edition, 2000.