A DESCRIPTIVE TITLE, NOT TOO GENERAL, NOT TOO LONG

Markus Püschel

Department of Computer Science
ETH Zurich, Switzerland

The hard page limit is 8 pages in this style. This excludes references and excluding the short, mandatory part on individual contributions (see end). Do not reduce font size or use other tricks to squeeze. This 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, why you do it (not necessarily in this order), and the main result. The abstract has to be self-contained and readable for a person in the general area. You should write the abstract last.

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.

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, real-time). Then explain that fast implementations are very hard and expensive to get (memory hierarchy, vector, parallel).

Contribution. Now you state what you do in this paper. In our example: presenting a DFT implementation that is faster for some sizes than all the other ones.

Related work. Next, you have to give a brief overview of related work. For a paper like this, anywhere between 2 and 8 references. Briefly explain what they do. In the end contrast to what you do to make now precisely clear what your contribution is.

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, self-contained summary of necessary background information on the algorithm or application that you then later optimize including a cost analysis.

For example, assume you present an implementation of FFT algorithms. You could organize into DFT definition, FFTs considered, and cost analysis. The goal of the background section is to make the paper self-contained for an audience as large as possible. As in every section you start with a very brief overview of the section. Here it could be as follows: In this section we formally define the discrete Fourier transform, introduce the algorithms we use and perform a cost analysis.

Discrete Fourier Transform. Precisely define the transform so I understand it even if I have never seen it before.

Fast Fourier Transforms. Explain the algorithm you use.

Cost Analysis. First define you cost measure (what you count) and then compute or determine on other ways the cost as explained in class. In the end you will likely consolidate it into one number (e.g., adds/mults/comparisons) but be aware of major imbalances as they affect the peak performance..

Also state what is known about the complexity (asymptotic usually) about your problem (including citations).

3. YOUR PROPOSED METHOD

Eval 3.6.

In the function Eval 3.6, we are given two matricies Q_m and P_m as input. Our goal is to find the Matrix R_m that solves the system of linear equations $Q_m * R_m = P_m$. If the input matrix A is an upper or lower triangular matrix, then so is the matrix Q_m. In this case we can directly solve the given system using forward substitution or backward substitution respectively. Both substitution methods have a runtime of $O(n^3)$. Since the rescaling phase at the end of the matrix exponential alorithm is different for triangular matricies, we have to perform this check at some point. If we do so before the function Eval 3.6 is called, we

can make this case distinction without any additional cost. In the case where Q_m is not triangular, there exist several different methods to obtain a solution to the given system of linear equations. Probably the easiest and fastest one would be to use Gaussian elimination in combination with backward substitution. This method tends to be numerically unstable for ill conditioned matricies. Since we cannot guarantee the conditioning of the input matricies and checking the properties of the matrix would add an significant overhead to the cost, we chose to use a numerically more stable algorithm right away. For this reason we decided to use an LUP-decomposition algorithm in combination with forward and backward substitution. We chose to use only partial pivoting instead of full pivoting since it is computationally less expensive and is said to have feasable numerical stability in practice. Our tests have verified this assumption. This gives us the following structure for the function Eval 3.6:

if(Q_m is lower triangular):
  R_m = forward-substitution(Q_m, P_m)
else if (Q_m is upper triangular):
  R_m = backward-substitution(Q_m, P_m)
else:
  L, U, P = LUP-decomposition(Q_m)
  P'_m = mmm(P, P_m)
  Y = forward-substitution(L, P'_m)
  R_m = backward-substitution(U, Y)

LUP-decomposition:. There are several different alogithms to compute the LUP-decomposition of a Matrix. We chose the recursive-right-looking-algorithm since it seems to be the one that could benefit the most from using SIMD-instructions. The algorithm has a runtime of $O(n^3)$ and performs the following steps:

1) Swap pivot-row with top-row
2) Partition matrix into (Piv, a_{12}, a_{21}, A_{22})

piv	a_{12}
a_{21}	A_{22}

3) Divide a_{21} by pivot
4) $A'_{22} = A_{22} - (a_{21} * a_{12})$
5) Continue at step one with matrix A'_{22}

Optimizations:. In step 3) of the LUP-decompostion we have to divide all elements of the vector a_{21} by the pivot. To avoid the costly division, we can compute the reciprocal of the pivot and then multiply all elements of a_{21} with it. Since we use coloumn-major format, the elements of a_{21} are aligned in memory. This enables us to use SIMD-instructions to perform the multiply on 4 elements simulatneously. In step 4) we have to subtract the outer product of two vectors from the submatrix A_{22}. This updating-step can be done using a doulbe-loop with the following formula:

$A'_{22}[i][j] = A_{22}[i][j] - a_{21}[i] * a_{12}[j]$

As the sub-matrix A_{22} is stored in column major format, and the elements of the vector a_{21} are stored in order, we can use SIMD-instructions to update 4 elements simultaneously.

Forward-substitution:. For a given lower-triangular Matrix L and a RHS Matrix B, we can find a solution Matrix Y by implementing the following formula using a triple-loop:

$y_{ij} = \frac{b_{ij} - \sum_{k=1}^{i-1} l_{ik} * y_{kj}}{a_{ii}}$

This formula shows that the computation of y_{ij} is dependent on $y_{(i-1)j}$. We therefore conclude that it is not possible to unroll the i-loop in a reasonable way.

Optimizations:. The triple loop can either be implemented using an $i - j - k$ loop order or using an $j - i - k$ order.

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 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 consider also icc for additional potential speedup.

Then explain what input you used and what range of sizes. The idea is to give enough information so the experiments are reproducible by somebody else on his or her

code.

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 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 the guide to benchmarking presented in class, in particular
- very readable, attractive plots (do 1 column, not 2 column plots for this class), proper readable 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).

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 (e.g., the results show that we are within 2x of the optimal performance ... Even though we only considered the DFT, our optimization techniques should be also applicable ....) You can also formulate next steps if you want. 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.

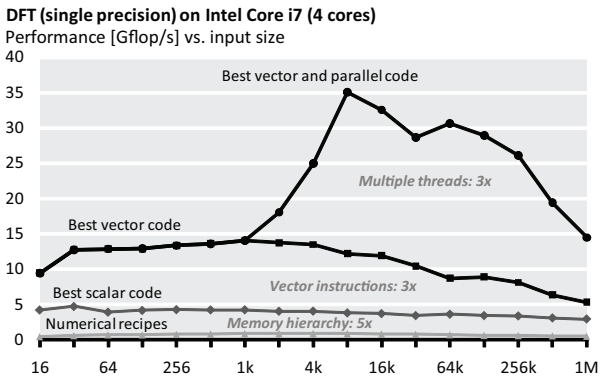

Fig. 1. Performance of four single-precision implementations of the discrete Fourier transform. The operations count is roughly the same. *The labels in this plot are about the smallest you should go.*

- 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. Nothing more awkward as when the story starts and one does not know what the direction is or the goal.
- 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 contributes to the grade.
- Books helping you to write better: [**?**] and [**?**].

Graphics. For plots that are not images *never* generate (even as intermediate step) jpeg, gif, bmp, tif. Use eps, which means encapsulate postscript, or pdf. This way it is scalable since it is a vector graphic description of your graph. E.g., from Matlab, you can export to eps or pdf.

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 8 pages.

7. CONTRIBUTIONS OF TEAM MEMBERS (MANDATORY)

In this mandatory section (which is not included in the 8 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 between one column and one page (absolute maximum).

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.

Do not include

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

Example and structure follows.

Marylin. 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 Cary and Jane on the SIMD optimization of variant 1, in particular implemented the bit-masking trick discussed.

Cary. ...

Gregory. ...

Jane. ...