A DESCRIPTIVE TITLE, NOT TOO GENERAL, NOT TOO LONG

Markus Püschel

Department of Computer Science
ETH Zürich
Zürich, Switzerland

The hard page limit is 6 pages in this style. 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.

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. It follows a possible structure of the introduction. Note that the structure can be modified, but the content should be the same. Introduction and abstract should fill at most the first page, better less.

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).

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: ELLIPTIC CURVE DIFFIE-HELLMAN

The purpose of this section is to familiarize the reader with the mathematical foundations of the Diffie-Hellman key exchange and which algorithms we used to achieve this task. This chapter follows a bottom-up approach.

Finite Field arithmetic [1, p. 3-4]. For elliptic curve cryptography one is mainly interested in the prime finite field $\mathbb{F}_p$ and the characteristic 2 finite field $\mathbb{F}_{2^m}$. In this paper only elliptic curves over $\mathbb{F}_p$ are discussed.

Montgomery multiplication.

Elliptic Curve over $\mathbb{F}_p$ [1, p. 6-7]. An elliptic curve is defined by the equation

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{1}$$

where p is an odd prime number and $a, b \in \mathbb{F}_p$ are the parameters of the curve. Furthermore a, b need to satisfy $4a^3 + 27b^2 \not\equiv 0$. The elliptic curve $E(\mathbb{F}_p)$ consists of the Points $P = (x, y)\ x, y \in \mathbb{F}_p$ satisfying (1). Additionally we introduce $\mathcal{O}$ the so called point at infinity. The addition of points (affine coordinates) is defined as follows [1]

1. $\mathcal{O} + \mathcal{O} = \mathcal{O}$
2. $(x, y) + \mathcal{O} = \mathcal{O} + (x, y) = (x, y) \quad \forall (x, y) \in E(\mathbb{F}_p)$
3. $(x, y) + (x, -y) = \mathcal{O} \quad \forall (x, y) \in E(\mathbb{F}_p)$
4. Assume $x_1 \neq x_2$. $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ is defined as $x_3 \equiv \lambda^2 - x_1 - x_2 \pmod{p}$, $y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p}$ and $\lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}$
5. $(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$ is defined as $x_3 \equiv \lambda^2 - 2x_1 \pmod{p}$, $\lambda(x_1 - x_3) - y_1 \pmod{p}$ and $\lambda \equiv \frac{3x_1^2 + a}{2y_1} \pmod{p}$

Rule 4. describes how to add two different points whereas rule 5. explains how to double a point. The set of points satisfying (1) and $\mathcal{O}$ together with this addition form a commutative group.

Jacobian Coordinates [2, p. 59 - 60]. "In cases where field inversions are significantly more expensive than multiplications, it is efficient to implement projective coordinates" [2] In Jacobian coordinates a point is represented as a triplet (x, y, z) satisfying (2).

$$y^2 \equiv x^3 + axz^4 + bz^6 \pmod{p} \quad (2)$$

In this project Jacobian coordinates were implemented. For the definition of the rules see [2, p. 59-60]

Double-and-add method. In order to implement the Diffie-Hellman key exchange we need to calculate dP fast.
Input $P \in E(\mathbb{F}_p)$, $d \in \mathbb{N}$
Output: $d \cdot P \in E(\mathbb{F}_p)$

```
N <- P
Q <- O
for i from 0 to m do
  if d_i = 1 then
    Q <- point_add(Q, N)
  N <- point_double(N)
return Q
```

Listing 1. double-and-add method

where $d = d_0 + d_1 2 + ... + d_m 2^m \quad d_i \in \{0, 1\}$

Diffie Hellman key exchange [3, p. 170-171]. Alice and Bob want to establish a secret over a public channel. We assume that the elliptic curve parameter: a prime p, $a, b \in \mathbb{F}_p$ a point G with high order are publicly known.

1. Alice chooses a secret integer u, computes $G_u = uG$, and sends G_u to Bob.
2. Bob chooses a secret integer v, computes $G_v = vG$, and sends G_v to Alice.
3. Alice computes $uG_v = uvG$
4. Bob computes $vG_u = vuG$.

One can verify that $uG_v = uvG = vuG = vG_u$. An eavesdropper knows G, G_u, G_v and his goal is to calculate uvG. This is known as the Diffie-Hellman Problem and is assumed to be a hard problem.

Complexity.

Cost measure.

3. YOUR PROPOSED METHOD

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 class, 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 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 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.
- 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.
- Make sure you define every acronym you use, no matter how convinced you are the reader knows it.
- Always spell-check before you submit (to me in this case).
- Be picky. When writing a paper you should always strive for very high quality. Many people may read it and the quality makes a big difference. In this class, the quality is part of the grade.
- Books helping you to write better: [4] and [5].
- Conversion to pdf (latex users only):

  dvips -o conference.ps -t letter -Ppdf -G0 conference.dvi

  and then

  ps2pdf conference.ps

Graphics. For plots that are not images *never* generate (even as intermediate step) jpeg, gif, bmp, tif. Use eps, which means encapsulate postscript, os 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.

Here is an example of how to get a plot into latex (Fig. **??**). Note that the text should not be any smaller than shown.

7. REFERENCES

[1] Daniel R. L. Brown, "Sec 1: Elliptic curve cryptography," http://www.secg.org/, 2009.

[2] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999, Cambridge Books Online.

[3] Lawrence C . Washington, *Elliptic Curves Number Theory and Cryptography*, Chapman and Hall/CRC, 2nd edition, 2008.

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

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