ESCAPE ANALYSIS FOR MULTI-THREADED JAVALI

Andrei Pârvu, Sebastian Wicki

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 the spacing 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 sorting implementation ever, one could roughly go as follows. First explain why sorting 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).

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

Related work. Next, you have to give a brief overview of related work. For a report 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.

2. BACKGROUND: WHATEVER THE BACKGROUND IS

Give a short, self-contained summary of necessary background information. For example, assume you present an implementation of sorting algorithms. You could organize into sorting definition, algorithms considered, and asymptotic runtime statements. 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 sorting problem we consider and introduce the algorithms we use including a cost analysis.

Sorting. Precisely define sorting problem you consider.

Sorting algorithms. Explain the algorithm you use including their costs.

As an aside, don't talk about "the complexity of the algorithm." It's incorrect, problems have a complexity, not algorithms.

3. YOUR PROPOSED METHOD

Now comes the "beef" of the report, 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.

In this section, structure is very important so one can follow the technical content.

Mention and cite any external resources that you used including libraries or other code.

4. EXPERIMENTAL RESULTS

Experimental setup. For the experiments we used two different machines. The first one, running Ubuntu 14.04 with a i5-3317U, 1.70GHz processor and 4Gb RAM.

We tried to analyze three different aspects of our project: firstly, the running time improvement of just allocating the unescaped objects on the stack. Secondly, we wanted to see the time benefit of working with stack-allocated objects

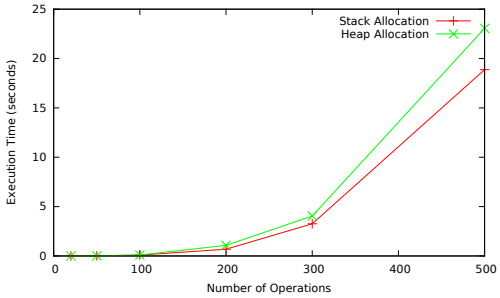

Fig. 1. Running times matrix multiplication

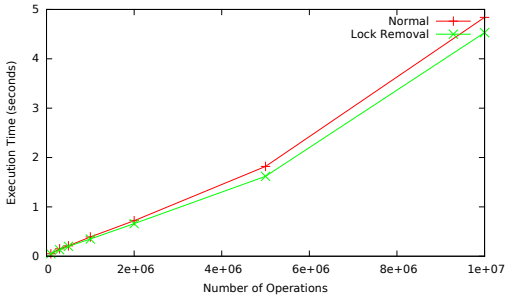

Fig. 2. Running times hash map

instead of heap-allocated ones. And thirdly, we tried to observe the how lock-removal affects single-threaded programs.

Results. For the first experiment, we just allocated various objects without using them afterwards, in order to see how the running time differs for both cases. TODO: run the actual experiment :)

In the second experiment, we wrote a basic matrix multiplication program with the intent of observing how execution time changes when using stack allocation. The results can be seen in Figure 1. It can be observed that the stack allocation improves running time with almost 20%, with the obvious draw-back that more stack memory is used.

For the third experiment, we implemented a hash map, which simulates string hashing (because Javali doesn't have strings we used ASCII arrays). The hash map is implemented to be thread-safe, so we ran the experiment with one thread and lock removal. Results can be seen in Figure 2; with lock removal, running time is approximately 10% faster than the unanalyzed version.

Comments:

- Create very readable, attractive plots (do 1 column, not 2 column plots for this report) with readable font size. However, the font size should also not be too large; typically it is smaller than the text font size. An example is in Fig. 3 (of course you can have a different style).
- Every plot answers a question. You state this question and extract the answer from the plot in its discussion.
- Every plot should be referenced and discussed.

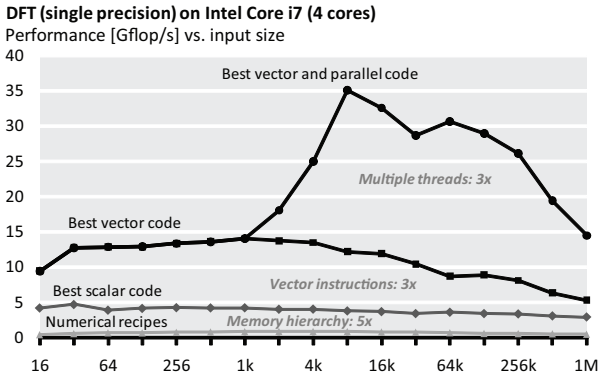

Fig. 3. Performance of four single precision implementations of the discrete Fourier transform. The operations count is roughly the same. The labels in this plot are maybe a little bit too small.

5. CONCLUSIONS

Here you need to 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 report, 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 such as high-level statements (e.g., we believe that .... is the right approach to .... Even though we only considered x, the .... technique should be applicable ....) You can also formulate next steps if you want. Be brief. After the conclusions there are only the references.

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 us 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: [**?**] and [**?**].
- 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 the bitmap formats jpeg, gif, bmp, tif. Use eps, which means encapsulate postscript. It is scalable since it is a vector graphic description of your graph. E.g., from Matlab, you can export to eps.

The format pdf is also fine for plots (you need pdflatex then), but only if the plot was never before in the format jpeg, gif, bmp, tif.

7. REFERENCES

[1] J. Whaley, M. Rinard, *Compositional Pointer and Escape Analysis for Java Programs* 1999

[2] E. Ruf, *Effective Synchronization Removal for Java* 2000

[3] V.P. Ranganath, J. Hatcliff *Pruning interference and ready dependences for slicing concurrent Java programs* 2004