Component-based Grid Programming
A Case Study on Wavelets

Jan Dünnebier & Sergei Gorlatch
University of Münster
Anne Benoit & Murray Cole
University of Edinburgh

Abstract

Grid middleware allows linking heterogeneous servers to internet-spanning computational grids. Component architectures can abstract over many technical arrangements and provide a higher-level interface to middleware. Thus, “Higher-order components” (HOCs) require from the developer application level programming only, and low level aspects are transparent to the developer. Similarly, high level parallel programming libraries allow the same abstraction for the developer, but these may be constrained by the choice of the target platform. For instance, the Edinburgh skeleton library (eSkel) can outperform component systems in some applications, but in the current version the target platform must have an MPI runtime environment, and the application must be written in C. HOCs in contrast can be deployed to any web service container and can even be implemented using multiple programming languages. In experiments on a grid-like testbed, we have applied a HOC for the “lifting scheme” algorithm to a wavelet transform on raster images. This application is integrated with eSkel and shows that the flexibility of a component can be combined with the performance of a native parallel library.

1 Introduction to Wavelets
Wavelet transforms are integral transforms, closely related to (windowed) fast Fourier transforms (FFT). While FFT decompose functions into sines and cosines, the discrete wavelet transform (DWT) projects functions onto a family of zero-mean functions (wavelets), such as the second derivative function of the Gaussian probability density function, whose graph is the commonly known "mexican hat" shown in fig. 1.

DWT can be implemented using the lifting scheme [1]. This report deals with a software component (the lifting-HOC) featuring a design similar to the lifting scheme, i.e. it is parameterised by other functions. This component can run DWT in parallel on top of a distributed computing environment. In the current work, we confine our attention to DWT via lifting. A broader overview of theoretical approaches to wavelets, e.g. via filter banks can be found in [2].

2 Challenges for Component Design
As the lifting scheme algorithm for DWT can be expressed using a higher-order function it is an ideal candidate for experimenting with frameworks like eSkel [4, 5] and HOC-SA [7].
The Edinburgh Skeleton Library (eSkel) is a C library of algorithmic skeletons [3] implementing recurring patterns of parallel behaviour such as pipeline and task deal. Using parameter functions, application specific code can be invoked within a parallel process. Technically eSkel's skeletons are built upon MPI, but only little MPI code must be written for an application, since the library abstracts most of the process communication details.

The HOC-Service Architecture (HOC-SA)
The Higher-Order Component Service Architecture is a runtime environment for Higher-Order Components [6] (HOCs) providing parallelism via partially implemented web services. Programmers upload application specific units of code to a code service that assigns them lookup keys. These keys can be used as parameters for other services that download the code units and embed them in parallel or distributed implementations.

The lifting-HOC
The lifting scheme algorithm can be structured in stages, which makes it eligible for a parallelisation using the pipeline skeleton in eSkel. This skeleton can coordinate control and data flow [9], when the lifting procedure is applied to multiple input data in parallel. A HOC can be used to wrap the skeleton into a service deployed to a high-performance host and might be accessed via HTTP from any internet client.

3 Project Status
Our implementation of the lifting-scheme algorithm as a higher-order component works for one and two dimensional input data. Our current research deals with a stage-skipping optimisation that explicitly instructs the skeleton to pass small-sized data records through all remaining pipeline stages, as soon as they have been processed. Experimentally, we observed that such customisations [8] help increasing the overall degree of parallelism in a system and thereby increase the performance of the application significantly.
This work has been performed under the Project HPC-EUROPA (RI3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 Structuring the European Research Area Programme and the EPSRC project Enhance (under grant number GR/S21770/01).

References