Extending MathWebSearch to a Distributed Environment

Corneliu C. Prodescu,
Supervisor: Michael Kohlhase
Jacobs University Bremen

January 30, 2012

Abstract

MathWebSearch is an open-source, open-format, content-oriented search engine for Mathematical formulae. It is a complete system capable of crawling, indexing and querying expressions based on their functional structure (operator tree) rather than their presentation. MathWebSearch indexes content-rich representation formats (like MathML) using substitution tree indexing, a data structure which shows great advantages in terms of space and time complexity. In this paper, we propose a distributed implementation of MathWebSearch, making it feasible for large scale applications.

1 Introduction

As the world of information technology grows, being able to quickly search data of interest becomes one of the most important tasks in any kind of environment, be it academic or not. MathWebSearch system is a search engine that addresses the problem of looking up mathematical formulae from a semantic point of view. Instead of performing a string-based search (as a standard engine like Google does), it does Mathematical content matching through unification. MathWebSearch harvests the web for Content MathML, the semantic flavor of MathML [ABC+03], which it indexes, storing the structure of each formulae in its core database.

1.1 History

MathWebSearch started off in 2006 as Ioan Sucan's Bachelor Thesis, under the supervision of Prof. Dr. Michael Kohlhase. The idea was to build a search engine which would look up Mathematical formulae based on structure, rather than presentation. After all, users searching for Mathematical formulas are interested only in semantics. The meaning-oriented goal suggested algorithms and data structures different from the typical string-oriented search engines. As such, a technique inspired from Automated Theorem Proving was brought in - the substitution indexing tree [Gra96]. The initial implementation is described in [KŞ06].

Between 2007 and 2010, the system was explored mainly on a theoretical level, in papers like [KK07]. Unfortunately, none of these concepts were featured into the system at the time.

In 2010, I re-implemented the system core, based on the same principles, but following a cleaner and more efficient structure. The main data structure was optimized, the service was de-coupled into separate components (crawler, term-indexing core and a RESTful interface) and the APIs were clearly defined. The current system is described in section 3.

1.2 Importance of Structure Awareness

In a search application, one of the most important features is the ability to clearly interpret a specified query. This becomes even more outstanding when the search engine is designed to deal with Mathematical formulae.

One of the issues that structure awareness solves is the inherent ambiguity in visual representations. Imagine the example of $f(a+b)$. Most search engines and text processing applications see this as a sequence of 6 characters. However, MathWebSearch, as a structure oriented system, sees one of the following:

- Application of function f on argument $a+b$[1]
- Multiplication of arguments f and $a+b$

Another important advantage is the ability to search up to alpha renaming. Imagine a query of the form

$$x^4 + 4y^4$$

In a typical text-based representation, this would have little connection with, for example,

$$m^4 + 4n^4$$

However, for MathWebSearch, the two formulae are equivalent, if we specify x, y as arbitrary query variables. The same reasoning can be used to search for Mathematical Theorems which fit a desired pattern.

2 State of the Art

There seem to be two general approaches to searching mathematical formulae. One generates string representations of mathematical formulae and uses conventional information retrieval methods, and the other leverages the structure inherent in content representations.

The first approach is utilized for the Digital Library of Mathematical Functions [MY03] and ActiveMath system [LM06]: mathematical formulae are

[1]Note that, in turn, $a+b$ is regarded as addition of arguments a and b

converted to text and indexed. The search string is similar to LaTeX commands and is converted to string before performing the search. This allows searching for normal text, as well as mathematical content simultaneously, but it cannot provide powerful mathematical search — for example, searching for something like $a^2 + c = 2a$, where a must be the same expression both times, cannot be performed. On the same topic, there is a system [MG08] which uses Reverse Polish Notation as intermediary format. These methods have the important advantage that they rely on already existing technologies, but they do not fully provide a mathematical formulae oriented search method.

The second approach is taken by the MBase system [KF01], which applies the pattern matching of the underlying programming language to search for OMDoc-encoded [Koh06] mathematical documents in the knowledge base. The search engine for the Helm project indexes structural meta-data gleaned from Content MathML representations for efficient retrieval [AS04]. The idea is that this metadata approximates the formula structure and can serve as a filter for very large term data bases. However, since the full structure of the formulae is lost, semantic equivalences like α-equivalence cannot be taken into account.

Another system that takes this second approach is described in [TSP06]. It uses term indexing for interfacing with Computer Algebra Systems while determining applicable algorithms in an automatically carried proof. This is closely related to what we present in this paper, the main difference being that we provide search for any formula in a predefined index, while in [TSP06] a predefined set of formulae characterizing an algorithm is automatically searched for in a changing index.

3 MathWebSearch-0.5 System

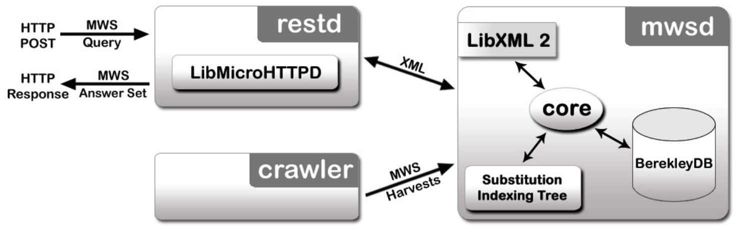

Figure 1: MWS-0.5 System Structure

In this section, we present an overview of the MathWebSearch System, as far as it is necessary to make this material self-contained. For a complete picture, as well as detailed benchmarks, see [PK11].

The MathWebSearch system consists of the three components pictured in Figure 1. The *crawler subsystem* collects data from MathML rich corpora, transforms the mathematical formulae into *MWS Harvest*s and feeds them into the core system. The *core system* (the MathWebSearch daemon mwsd) builds the search index and processes search queries: it accepts the MathWebSearch input formats (*MWS Harvest* and *MWS Query*) and generates the MathWebSearch output format (*MWS Answer Set*). These are communicated through the *RESTful interface* restd which provides a public HTTP API conforming to the REST paradigm [FT02].

These components have been implemented using POSIX-compliant [POS88] C^{++}. We use the MicroHTTPd library [Mic] API for handling HTTP, and LibXML2 [Vei] API for XML parsing. The meta-data accompanying the internal index is stored using an external database system. As we are dealing mainly with key-value pairs retrieval, the BerkeleyDB [Ber09] API was preferred.

For the purpose of this proposal, we will focus on the core application, the indexing structure and the current workload capabilities.

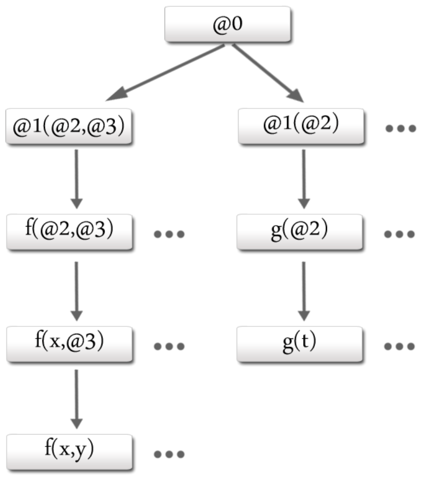

Figure 2: Depth-First Substitution Indexing Tree

As previously mentioned, the main data structure is a substitution indexing tree (see figure 3). The root consists of the most generic term (denoted @0) and, as we traverse the tree, each parent-child link specifies a substitution. To avoid having multiple paths for the same instantiated expression, the substitutions are always represented in depth-first order.

This model allows structural awareness, as well as great retrieval performance. On the downside, subterms[2] cannot be easily fetched, so indexing a term implies indexing all its (distinct) subterms.

The current system is capable of handling harvests of around 20 million expressions, with acceptable query times (under 100 ms). Of course, these statistics depend on the run-time environment.

[2]Subterms of a term are the terms described by subtrees of the main operator tree. For example, the subterms of $f(g(x), y)$ are $g(x)$, x and y.

4 Proposal Outline

4.1 Objective and Challenges

One of the envisioned use cases of MathWebSearch is indexing the Cornell ePrint archive (See `http://www.arXiv.org`). The MathML extracted from the 700,000 TEX/LATEX articles is estimated to contain an order of 10^8 Mathematical formulae. Including subterms, this number of terms to be indexed will reach at least 10^9.

Assuming the average term can be encoded using only 64 bytes, we would still need 64 Gbytes of RAM to index all the formulae. As this seems unfeasible for a single machine, the distributed approach idea emerged.

A distributed environment implies a distribution framework (communication architecture, data marshaling, caching, etc) and a distribution policy. In most applications, a simple hashing policy performs reasonably well, as it scatters the data uniformly. However, such approaches are problematic with MathWebSearch:

- Node-scattering: breaking the index tree across machines is very inefficient, as cross-machine indirection[3] implies huge overheads (memory read vs network access).
- Expression-scattering: breaking the harvests and building multiple trees, one on each machine. While hashing the harvests and diving the data uniformly is possible, hashing the queries is irrelevant. Hence, queries would have to be sent to every machines and the results should be merged constantly.

4.2 Proposed Design

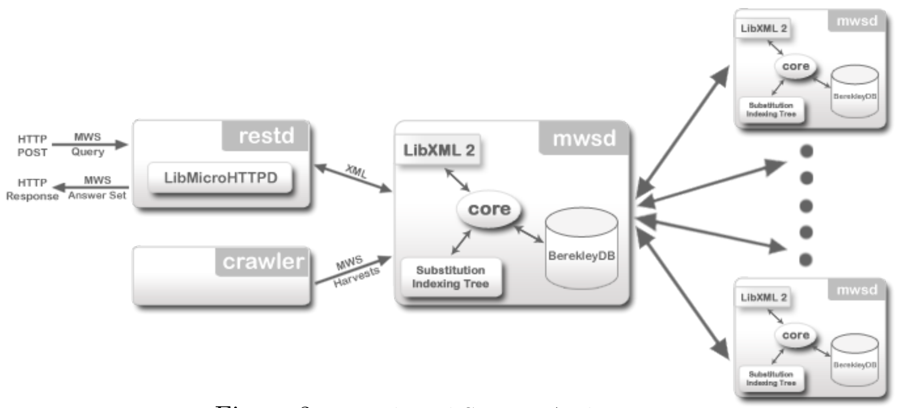

Figure 3: Distributed System Architecture

[3]When a parent node is on machine A and a child node is on machine B

We will start with a simple distribution framework, supporting master-slave architecture, as presented in figure 3.

We propose a hybrid distribution pattern; we will partition the main tree into subtrees and scatter these across multiple machines. For reasonable size subtrees (in the Gb range), the cross-machine indirection problem is fixed, as queries will make very few jumps. Also, it is important that the design remains simple and generic, as each subtree is a substitution indexing tree with a different initial instantiation. An example subtree partitioning is given in figure 4

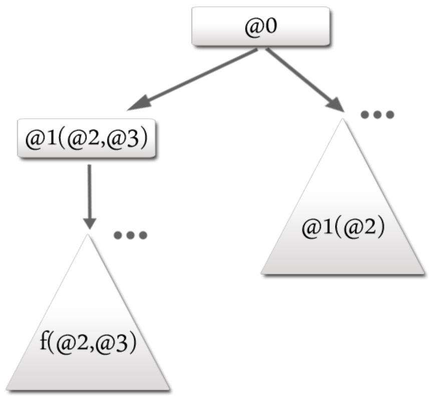

Figure 4: Subtree partitioning

This approach has a great scalability potential, as subtrees can be dynamically created and migrated across machines. Hence, automatic space and load balancing can be implemented. The exact thresholds will be determined as the features are implemented, the trade-off being between temporary migration overhead vs slightly uneven utilization.

5 Conclusions and Timeline

Following the ideas presented here, implementing a distributed MathWebSearch System seems feasible.

The system will be evaluated on a computer cluster, in configurations ranging from 3 to 6 machines and index sizes of up to 10^8 expressions. The end result will be successful if the query times will remain under 100ms and the system will scale at most linearly[4] in the tested configurations.

The expected timeline for the project is outlined next. The detailed Gantt chart is presented in figure 5.

- 1st Feb 2012 - 15th Feb 2012: Design
- 16th Feb 2012 - 21st Apr 2012: Implementation

[4]This refers to the index size vs number of machines graph.

- 22nd Apr 2012 - 6th May: Configuration and Testing
- 1st Apr 2012 - 18th May: Thesis writeup and Presentation

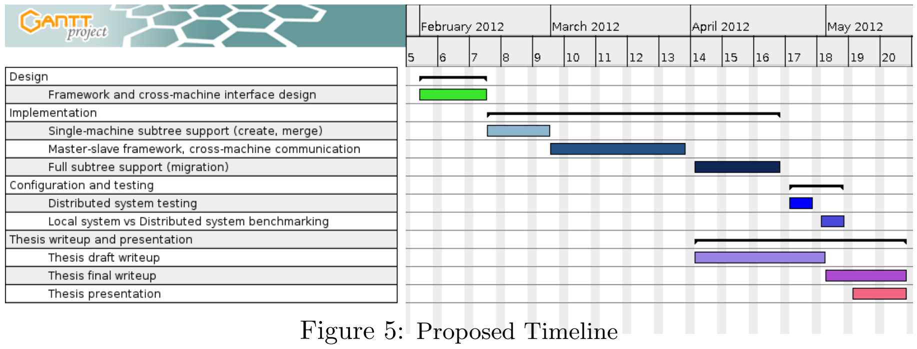

Figure 5: Proposed Timeline

References

[ABC$^+$03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium (W3C), 2003.

[AS04] Andrea Asperti and Matteo Selmi. Efficient retrieval of mathematical statements. In Andrea Asperti, Grzegorz Bancerek, and Andrej Trybulec, editors, *Mathematical Knowledge Management, MKM'04*, number 3119 in LNAI, pages 1–4. Springer Verlag, 2004.

[Ber09] Berkeley DB. available at `http://www.oracle.com/technology/products/berkeley-db/`, 2009. seen January.

[FT02] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2, May 2002.

[Gra96] Peter Graf. *Term Indexing*. Number 1053 in LNCS. Springer Verlag, 1996.

[KF01] Michael Kohlhase and Andreas Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation; Special Issue on the Integration of Computer Algebra and Deduction Systems*, 32(4):365–402, 2001.

[KK07] Andrea Kohlhase and Michael Kohlhase. *Re*examining the MKM Value Proposition: From Math Web Search to Math Web *Re*Search. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants. MKM/Calculemus*, number 4573 in LNAI, pages 266–279. Springer Verlag, 2007.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, August 2006.

[KŞ06] Michael Kohlhase and Ioan Şucan. A search engine for mathematical formulae. In Tetsuo Ida, Jacques Calmet, and Dongming Wang, editors, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, number 4120 in LNAI, pages 241–253. Springer Verlag, 2006.

[LM06] Paul Libbrecht and Erica Melis. Methods for Access and Retrieval of Mathematical Content in ActiveMath. In N. Takayama and A. Iglesias, editors, *Proceedings of ICMS-2006*, number 4151 in LNAI. Springer Verlag, 2006. `http://www.activemath.org/publications/Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006.pdf`.

[MG08] J. Misutka and L. Galambos. Mathematical extension of full text search engine indexer. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1 –6, april 2008.

[Mic] GNU MicroHTTPd library. `http://www.gnu.org/software/libmicrohttpd/`. seen Jul 2011.

[MY03] Bruce R. Miller and Abdou Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1-3):121–136, 2003.

[PK11] Corneliu C. Prodescu and Michael Kohlhase. MathWebSearch 0.5 - Open Formula Search Engine. sep 2011.

[POS88] IEEE POSIX, 1988. ISO/IEC 9945.

[TSP06] Frank Theiß, Volker Sorge, and Martin Pollet. Interfacing to computer algebra via term indexing. In Silvio Ranise and Roberto Sebastiani, editors, *Proceedings of the 13th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus-2006)*, 2006.

[Vei] Daniel Veillard. The XML c parser and toolkit of gnome; libxml. System Home page at `http://xmlsoft.org`.