DELPH-IN$_{21}$

LANGUAGE TECHNOLOGY IN THE 21ST CENTURY

— Application for Advanced User Support (March 2, 2009) —

1 Details of the Applicant

Name	Stephan Oepen
Position	professor
Affiliation	Universitetet i Oslo, Institutt for Informatikk
Address	Boks 1080 Blindern, 0316 Oslo
Telephone	(+47) 2284 0125
Email	oe@ifi.uio.no

2 NoTur Project Account

This is a first-time application to NoTur from the UiO Language Technology Research Group (`http://www.ifi.uio.no/research/groups/lns/lt.html`), hence there is no existing project account. The group is submitting a parallel application for cpu allocations (of 20,000 units), however, in the category of 'new' projects.

3 Project Title and Summary

Title DELPH-IN$_{21}$ — Language Technology in the 21st Century

Summary Standard tools in language technology have matured to a point that faciliates the in-depth analysis of relatively large amounts of running text. The DELPH-IN (`http://www.delph-in.net`) open-source repository provides analysis tools for a variety of languages, developed and applied collaboratively by a network of sites world-wide. The existing technology is not yet ready for deployment on large-scale HPC resources. This project will adapt the DELPH-IN tools for distributed and parallel processing, including some (re-)packaging for smooth operation with standard job management systems, completion of existing but partial MPI support, and conversion of older code from PVM to MPI.

4 Objectives

The DELPH-IN open-source repository of language technology tools provides technology for (a) *parsing* (the analysis of text into an abstract representation of meaning), (b) *generation* (the inverse process: 'verbalizing' a meaning representation), (c) *translation* (as the combination of parsing and generation, combined with a cross-linguistic step of *transfer*), and (d) *machine learning* (through the estimation of probabilistic models that guide the search for solutions in processes (a) to (c)). This technology has been under continuous development since the early 1990s and has a world-wide user community of a few hundred researchers (and one known commercial application). The project applicant has been a contributor to this development since around 1994.

At present, the DELPH-IN tools are not immediately applicable to large-scale HPC environments. This is in part owed to missing 'packaging' and interfaces (for effective batch operation), and in part owed to a lack

of support for distributed and parallel processing. For processes (a) through (c) there is software support for parallelization (at a relatively crude level, viz. processing of one complete sentence), but it is technologically out-of-date (based on PVM) and severely limited in scalability (due to very large messages and naïve, serial scheduling). Therefore, the current technology cannot take advantage of more than a handful of cpus. Process (d)—the machine learning component that operates 'on top' of parsing, generation, or translation—is maybe the most resource-intensive task. At present, training of a probabilistic model cannot be parallelized, even though the toolkit used most for this task (see § 5 below) already includes partial support for use with MPI. Finally, the DELPH-IN developer community has quite limited HPC experience to date.

The main objectives of the proposed project are to (a) adapt the core DELPH-IN tools for effective use on the UiO TITAN cluster and (b) to train the UiO members of DELPH-IN in the daily routines of HPC usage. Specifically, the project will (i) study and analyze common use patterns of DELPH-IN tools, (ii) adapt or develop 'wrapper' scripts (suitable for batch operation) for standard tasks, (iii) complete the dormant MPI support in the machine learning component and enable parallel execution of process (d) above, (iv) profile the two most time-critical tools (processes (a) and (d) above) in-depth and analyze bottlenecks in their execution on TITAN (both tasks are relatively memory-intensive, with core sizes of between two and twenty gigabytes), and (v) prepare the migration from PVM to MPI for processes (a) to (c) above, including a possible re-design of communication and schedulig strategies.

5 Software

Following is a list of components from the DELPH-IN repository that will be targeted in the current project. All operate on linguistic resources (i.e. computational grammars of various natural languages) developed in the so-called DELPH-IN Reference Formalism, a description language for so-called unification-based (or constraint-based) grammar engineering, in a sense a specialized programming language for linguists. The DELPH-IN formalism implements the logic of typed feature structures, with multiple inheritance and unification as its central elements. Loosely speaking, the DELPH-IN formalism is a generalization of a logic programming language like Prolog: adding typing, multiple inheritance, and graph (rather than term) unfication. The proposed project will take the so-called LOGON tree (`http://wiki.delph-in.net/moin/LogonTop`) as its point of departure, essentially a DELPH-IN 'distribution' developed and maintained at UiO.

Parsing PET (Platform for Efficient Experimentation with HPSG Processing Techniques; `http://www.delph-in.net/pet`) is the egine used commonly for batch parsing with DELPH-IN grammars: given a sequence of natural language utterances (typically sentences), it performs an n-best search for syntactic and semantic analyses according to the constraints provided by the grammar. PET is implemented in C[++] (with core routines in pure C) and was originally developed in the mid-1990 by Ulrich Callmeier (then an MSC student at Saarland University, Germany; supervised by the project applicant). At the time, PET was relatively carefully optimized, down to levels of memory locality, register utilization, alignment, and instruction pipelining (all targeting 'large-memory' UltraSparc hardware of those days). PET was subsequently 'hardened' for industrial use (at YY Software, a Silicon Valley start-up of which the applicant was a principal), and since around 2002 is maintained by a group of developers (Peter Adolphs and Bernd Kiefer, German Research Center in AI; Stephan Oepen, UiO; and Yi Zhang, Saarland University). The software has seen substantial enhancements in functionality in recent years, but for almost ten years it has not been carefully profiled and optimized for maximum effiency on modern hardware. At the same time, DELPH-IN grammars have grown substantially in recent years, and the technology is now more standardly applied to relatively large amounts of unrestricted, running text. When parsing Wikipedia, for example, parse times per sentence (each of about 18 words in length) average at just below five seconds, which for the full Wikipedia would amount to about six (serial) cpu years. A semantic search enterprise like PowerSet (`http://www.powerset.com`), however, needs to turn around this process approximately once each month.

Generation The LKB (Linguistic Knowledge Builder; `http://www.delph-in.net/lkb`) is another processing engine that implements the DELPH-IN formalism. The LKB is primarily designed for interactive use, i.e. it serves as the development environment for grammarians (including a variety of graphical interfaces and debugging and tracing support). Unlike PET, the LKB provides support for generation (the inverse of parsing), and it is thus standardly used for batch processing too. The LKB is implemented in Common-Lisp, where

the commercial Franz Allegro CL is required for the full LKB functionality. The LKB was originally developed by Ann Copestake (at the University of Cambridge, UK), and in the late 1990s was completely re-factored in a joint effort of Ann Copestake, John Carroll (University of Sussex, UK), Rob Malouf (Stanford University, USA), and Stephan Oepen (then at Saarland University). Except for Rob, the three original developers actively maintain the software today. Given its frequent use in teaching—in addition to language technology research discussed presently—the LKB presumably has a few thousand users (at a few hundred sites) world-wide. Like PET, the LKB currently provides a PVM-based API to the so-called [incr tsdb()] 'profiler' (see below), and its use for batch generation will serve to exemplify the use of Lisp-based clients in the DELPH-IN pipeline). Generation using the LKB can be very memory-intensive, and the project will perform a limited amount of investigation and adaptation of Lisp-internal parameters (mostly controlling allocation strategies and garbage collection), aiming to allow effective use of the LKB on standard TITAN nodes.

Machine Learning TADM (Toolkit for Advanced Discriminative Modeling; `http://tadm.sf.net`) is the main machine learning component used in DELPH-IN research to date. TADM estimates the parameters of so-called discriminative, log-linear (or exponential) statistical models, where the result of this process can subsequently serve to probabilistically rank competing hypotheses in one of the search processes (a) to (c) above, say direct the parser towards the most probable syntactic and semantic analysis. TADM is implemented in C$^{[++]}$, built on top of the PETSc and TAO libraries, and was originally developed by Rob Malouf (then at the University of Groningen, The Netherlands). A group of active TADM users, collaborating with Rob, hosted the project at SourceForge around 2004 and consolidated existing patches (including some from users at UiO). Otherwise, there has been no active TADM development in recent years, and available documentation is sparse. TADM is applied to training data (typically in the form of millions or billions of integer-coded 'features') prepared using the [incr tsdb()] software (see below), and a single estimation run can take several cpu hours. In searching for best-performing model parameters, dozens or hundreds of distinct configurations need to be tested, typically each by means of ten-fold cross validation. Hence, in current development, TADM throughput is the primary bottleneck. Reportedly, a parallel version of TADM was available locally at Groningen in the late 1990s (customized for MPICH and Myrinet), and the proposed project will resurrect (and adapt as needed, for use on TITAN) MPI support in TADM. Also, it will be necessary to profile some of the core routines and experiment with different versions of low-level libraries (notably BLAS and LAPACK) and use of the Intel compiler suite (rather than the vanilla GNU Compiler Collection), to further improve the cpu utilization of TADM.

Distribution and Parallelization The [incr tsdb()] (`http://www.delph-in.net/itsdb`) environment provides 'profiling' capabilities for DELPH-IN grammars and software. System inputs (test suites and text corpora) are maintained in the [incr tsdb()] database, [incr tsdb()] controls the execution (and distribution) of batch processing, and processing results are recorded in the same database for subsequent analysis. [incr tsdb()] is comprised of a relational database layer (implemented in ANSI C), the [incr tsdb()] kernel (in Common-Lisp, interfacing to the PVM C binding), and a graphical user interface (in Tcl/Tk). The software has been continuously developed by Stephan Oepen since the mid-1990s, with contributions by numerous others, and today acts as a kind of 'clearing house' across other DELPH-IN components. Using PVM facilities, [incr tsdb()] allows distributed batch processing (with either PET or the LKB) and provides basic job management, failure detection, and roll-over functionality. However, non-trivial communication overhead, the naïve approach to task scheduling (serial, round-robin), and limitations in the database backend today limit the scalability of batch processing combining PET and [incr tsdb()]. In preliminary experiments on TITAN, it appears impossible to let the system take advantage of more than about ten cpus, i.e. the current scheduler does not scale very well. Furthermore, [incr tsdb()] is used in pre-processing training data for TADM (extracting and encoding a broad range of features for the machine learner) and for the evaluation of different feature and (estimation) hyper-parameter configurations. This part of [incr tsdb()], at present, has no support for distribution or parallelization.

6 Future Use of the Application Software

All components described in § 5 above are free software and essential parts of the daily work cycle of most DELPH-IN users. The Language Technology Research Group at UiO uses DELPH-IN resources as the point of departure in the vast majority of externally funded research as well as in student and doctoral projects (see `http://www.emmtee.net` and `http://www.velldal.net`, for example). Enabling the group

to make more effective use of local and national HPC facilities will be a key factor in scaling up this line of research to continuously growing target corpora (with a vision of 'web-scale' semantic parsing) and more complex linguistic and mathematical models. Both are strong international trends in contemporary language technology research, where 'empiricist' or 'data-driven' approaches have gained a dominant position. Access to large-scale compute facilities will enable the group to compete successfully for premium research results internationally—few conference papers in language technology today are accepted for publication without an extensive empirical evaluation—and it will further strengthen the role of the UiO team in the world-wide DELPH-IN consortium.

7 Problem Description and Expected Measurable Results

The summary of targeted software (see § 5 above) already gave an indication of current limitations in the DELPH-IN technology. The UiO Language Technology group used to maintain a 'private' mini-HPC cluster (for a total of 16 cores, 56 gbytes of physical memory, and some 400 gbytes of disk space), but these resources are no longer sufficient to execute state-of-the-art research in language technology. Given strong trends towards larger (input) data sets to be processed and more cpu- and memory-intensive approaches, the group strongly believes that it can only obtain the scalability properties required for this kind of work through access to genuinely large-scale (and constantly improving) HPC resources.

Individual group members have enjoyed access to TITAN since around 2007, essentially as a trial period to prepare the full-scale transition. From this limited experience, both the utility of centralized, shared HPC facilities, as well as remaining obstacles and limitations in the current tools are already evident. In the past, a lot of the workload executed by group members on TITAN was initiated through interactive sessions (i.e. requesting one or more nodes exclusively from the scheduling system). This approach is cumbersome, inefficient, and ultimately not scalable. Furthermore, at times of higher load on TITAN, it can be impossible to gain access to cpu resources this way.

As measurable results of the proposed project, it is expected that objectives (i) through (v) as detailed in § 4 above will be met. Upon completion of this project, parsing, generation, and machine learning using DELPH-IN technology shall be straightforward tasks to be executed through the TITAN queue system, where all members of the UiO Language Technology group (see § 9 below) have access to TITAN and the knowledge to perform non-trivial experimentation without additional hand-holding. Modified versions of PET and the LKB will be available that are optimized better for the AMD64 architecture and standard memory configurations on TITAN. Furthermore, it will be possible to parallelize machine learning using TADM (i.e. individual estimation runs) and to distribute multiple such runs from the [incr tsdb()] controller. The greatly increased throughput in experimentation will enable a surge in large-scale, data-driven research on language technology at UiO.

8 Need for Assistance

The UiO Language Technology group comprises internationally renowned language technology competency and experience, but the group at present lacks in-depth HPC knowledge. The effective utilization of TITAN facilities—including its scheduling system, optimum strategy for running memory-intensive tasks on 'standard' nodes, use of MPI, and experimentation with the available large-memory nodes—are all tasks that require complementary, in-depth technological expertise and close (and sustained) collaboration with TITAN staff. Given the comparatively large memory footprint of some of the DELPH-IN tools, finding the best balance of cpu and memory utilization (both and 'standard' and 'fat' nodes) will be a central factor in this project

9 Participants from User Side

The project applicant is the head of the UiO Language Technology group and will be the primary point of contact to the TITAN group. As the main developer and co-maintainer of most of the software involved in the project, the applicant will participate actively in the run-time analysis, software adaptation, and (re-)packaging of DELPH-IN tools. He will also serve as the interface to the larger international DELPH-IN community, drawing in additional test users as appropriate. Two current doctoral researchers (Gordana Ilić Holen and Gisle Ytrestøl) pursue PhD projects, based on the DELPH-IN repository, that will demand large-scale computation during the

proposed project duraction. However, the role of these PhD researchers will be more as 'standard' TITAN users, not necessarily developers in the project proposed presently. Finally, the group expects to hire a new post-doctoral fellow (possibly Erik Velldal) later in 2009, who will then complement the project applicant at the in-depth technical level.

10 Participants from Support Side

The current proposal was prepared in consultation with the USIT group for Research Computing Services (VD: *Vitenskapelig Databehandling*) at UiO. VD intends to allocate two of their senior staff members to this project, Dr. Simen Gaure and Dr. Thierry Toutain. Gaure is a long-term member of the VD group and an experienced programmer who has detailed knowledge about the job scheduler system and available hardware on TITAN. Dr. Thierry Toutain is a new hire to the group, who will start his appointment on position April 1. Toutain is a very experienced programmer as well, and he will gradually take over the support functions for this project as he gains TITAN-specific experience under the guidance of Gaure and Dr. Hans Eide, the VD group leader.

11 Project Plan

Given the 'starter package' nature of the proposed project, and the fact that one of the project participants will only start work at UiO in April, the project plan for the time being has a deliberate 'shopping list' character. The project will comprise five work packages, corresponding to tasks (i) to (v) described in § 4 above. The first two work packages have a foundational nature; they need to be carried out sequentially and should be more or less complete before the remaining work packages can start.

(I) Use Analysis For the first few weeks of the project, participants from both groups will establish a shared knowledge of the relevant DELPH-IN tools, standard tasks and use patterns, and their interaction with the HPC environment. This work package will be organized as a series of hands-on 'walk-through' sessions, initially at a rate of about two half days per week. The main result of this phase in the project will be a mutual, in-depth understanding of the available resources, requirements, and limitations in current software versions. This knowledge will be documented through a collection of collaboratively authored wiki pags.

(II) Packaging Based on the observations made in the initial project phase, VD and user staff will jointly adapt and 'package' DELPH-IN tools for batch processing, i.e. prepare a gradual shift of use patterns, away from the currently predominant interactive style. This work package will predominantly provide a standard set of scripts and configuration mechanisms that automate common tasks and provide a good balance between ease of use and flexibility, i.e. user customization of relevant parameters. The resources developed in this work package will be contributed as new open-source components to the DELPH-IN repository.

The remaining three work packages target individual components of the DELPH-IN repository and ultimately aim for (albeit moderate) code modifications in these packages. These tasks could be executed more or less in any order, or even in parallel. Seeing the small number of people involved in the project, however, and the need for tight synchronization across the two groups, it is expected that also work packages (iii) to (v) will be implemented mostly sequentially.

(III) MPI Support in TADM In order to eliminate the current biggest bottleneck in the DELPH-IN toolchain, turn-around times in machine learning experiments need to be reduced substantially. This work package will be predominantly implemented by VD staff, (re-)enabling the incomplete and currently dormant MPI support in the TADM code base. Once the software modifications are complete, a joint series of experiments of increasing complexity will serve to determine the scalability of the TADM core (numeric optimization, processing huge sparse matrices). The extended TADM software will be integrated with the LOGON tree and contributed to the TADM project repository at SourceForge.

(IV) Application Profiling Both PET and TADM batch runs can quickly grow to several gigabytes in process size. Yet, the Lisp-based LKB and [incr tsdb()] processes demand even larger core sizes, ranging from around five to over thirty gigabytes. Running on 'standard' TITAN nodes typically means that only a subset of cores can be utilized. From cursory, non-systematic observations made so far, it is also evident that saturation of the memory sub-system can be a limiting factor. The project partners—with VD as the technical lead—will jointly apply various application-level and in-kernel profiling tools to better analyze the exact nature of these limitations.[1] This work package will optimize the memory footprint of PET, TADM, and the LKB, search for code improvements in critical parts, and generally identify the best-performing strategies for deployment on TITAN. Thus, this part of the project critically depends on the availability of large-memory nodes (with 128 gbytes or RAM and more). Documentation of these routines will be provided as part of the project wiki.

(V) TADM Distribution While work package (iii) will help speed up the completion of a single TADM run, a typical experiment may involve many hundred such runs. The [incr tsdb()] feature management wrapper around TADM applies two levels of caching (one BDB-based, for low-level feature information; the other filesystem-based, at the level of complete training instances), where all experiments share the low-level cache, and related sub-sets of experiments re-use the higher level cache. At present, the only option for parallel execution of experiments is to manually group related sub-experiments, create the intermediate caches, and then independently launch groups of batches. For optimum interaction with the caching mechanisms, such experiment-level control should be implemented within [incr tsdb()] proper. The scheduling and parallel execution of individual TADM runs should then be accomplished through MPI, although it remains to be determined in which exact way TADM results (very large parameter vectors) shall be communicated back to the [incr tsdb()] controller.

Besides the software extensions and modifications (and associated documentation) described above, the project will produce three reports, viz. (a) a detailed work plan and schedule, due six weeks after the start of the project; (b) a half-way progress report in the seventh month of the project; and (c) a final report, due one month after project completion.

12 Resources

The proposed project is for a duration of twelve months, with a start date of May 1, 2009. The USIT VD group will allocate, on average, one third of a full-time senior analyst—i.e. about a day and a half per week—for a total of four person months or 650 person hours. Because the project applicant and members of the Language Technology group will need to experiment on their own ('practicing', in a sense) parallel to working with VD staff, a relatively low-key part-time involvement of VD staff seems the most cost-effective way of achieving the project goals. There will be some high-activity periods, when member from both groups work jointly, for example for parts of work packages (iii) and (iv). Other parts of these work packages (some profiling and code modification) can be carried out by VD staff working more independently, and here too it is expected that during such periods participants will use substantially more than twelve hours per week on the project.

CPU allocations for the experimentation covered by this proposal are sought in parallel, through a separate application in the NoTur 'new' projects category.

13 Publication

The proposed project is an 'enabling' activity, aiming to lead the UiO language technology group into routine usage of TITAN (and potentially also other national HPC facilities). Undoubtedly, the increased scalability of DELPH-IN technology and shorter turn-around time for experiments will contribute substantially to the research activities of all group members and, thus, also lead to high-quality publications.

[1] PET, for example, used to (optionally include) custom mmap(2)-based memory management facilities, which currently remain disabled for portability reasons. The project will aim to eliminate these obstacles.