Mineralized substrate
Any point can be chosen to investigate the materials provided by contemporary AI
(Deleuze and Guattari (1976)) and the hardware level is just another entrance. This
chapter investigates one of the most basic levels of AI: the hardware. Digging deeper
we find electrons moving around, but this level usually does not make a lot of sense
from the perspective of AI. It is important to keep in mind that also software is in the
end matter and energy moving around.

The computer is not a new invention. Actually, it is a very old one and goes back
to at least the Sumerians (2000 BC), who used human computers to calculate taxes
and important astronomical data. The ancient Greek used humans to calculate the size
and shape of the earth. Napoleon Bonaparte had a factory floor of about 80 human
computers to calculate artillery tables for different weather conditions and to assist in
the change from the old imperial to the new metric system. Later Babbage created
the Difference Engine in 1832, which was probably the first mechanical computer. It
took until the first half of the twentieth century to do electronic analog computations
(actually approximations) of differential equations. The first electronic digital system
arrived in 1945, hailing the advent of the digital age. The computer architecture devel-
oped in the last part of the second World War is still dominant, although new paradigms
such as cloud and quantum computing are being tracked at the moment.

The present day state-of-the-art hardware consists of integrated circuits created out
of metal(-like) materials such as germanium, silicon, copper and the like. They are
created to structure the flow and regulate the position of electrons. The hardware is the
mineralized substrate on which the electrons flow, the mineral substrate also controls
the flow. The idea behind the system is not unlike irrigation systems. Just as with
irrigations systems, having a poor knowledge of the complete system can wreak havoc.

The metaphor of mineralization might seem a bit odd. The mathematical mental pro-
dcedures developed in the last couple of millennia slowly mineralized into matter. At
first it might have been wooden sticks or clay balls which could represent a physicality
such as the amount of sheep in a herd. Later clay tablets, papyrus scrolls and pen and
paper were used which were already an abstraction of the mental methods. Even later
calculation devices were created, ranging from the Chinese abacus to the difference
engine of Charles Babbage, from the analog computer to contemporary super comput-
ers. The incremental buildup of layer upon layer can be seen as the mineralization of
the mental mathematical capabilities into a complex physical device. The difference
instances of the mineralized machinery can be interpreted as a abstract search machine.
with phase transitions.

In the following paper (van der Zant (2008)) the hardware is being pushed out of its stable state to its limits. By trying to increase the amount of electrons flowing through the system it becomes clear that the hardware shows little self-organizing properties. This is not unexpected. The only way to increase the electron flow is to keep finding solutions to the bottlenecks that occur. The problem-solving processes do not seem to lead to singularities. This does not imply that hardware cannot show bifurcations, just that contemporary hardware does not seem to be able to do so.

Preferably most scientist use software to abstract away from the hardware. Tools and processes have been used that rely on the possession of a single integrated circuit. Current trends toward multi-core computers might be a signal that some of the methods taken for granted in the last decades do not function that well anymore. If a process can only be calculated in a linear manner on a single core, then the use of a multi-core or distributed computer will not help. In the case of GAI, the multi-core computer is preferable in order to generate many structures for the sorting machine. These ripples at the bottom of the way humans operate computers might be a signal that there is a singularity on the horizon. Using these new computer systems requires a new way of working, a new way of thinking, and a new way of programming. Perhaps cloud computing will allow for bifurcations, if programmed correctly.

Current trends in computer programming rely on the use of threads, which are pieces of the software that function more or less autonomous. Typically the interface (buttons, mouse interactions, . . . ) and the engine (the connection to the Internet and the rendering of the web page, for example) run in separate threads. This methodology stems from linear thinking, and scales only to systems with a few cores. But these methods do not scale to a 1024 core, or a 65536 core computer, since only a few threads are required.

Instead of thinking linear, what is required is to start thinking dynamical and non-linear. Instead of trying to keep the software under human control (which it already is not in many cases), methods have to be developed where the software organizes itself. Perhaps for self organization to happen the hardware might also have to evolve further. At least there are already very powerful multi-processor systems available to experiment with non-linearity on the software level.

Building a system that allows to search a very large database of document images requires professionalization of hardware and software, e-science and web access. In
astrophysics there is ample experience dealing with large data sets due to an increasing number of measurement instruments. The problem of digitization of historical documents of the Dutch cultural heritage is a similar problem. This paper discusses the use of a system developed with the Kapteyn Institute of Astrophysics in Groningen, for the processing of large data sets, applied to the problem of creating a very large searchable archive of connected cursive handwritten texts. The system is adapted to the specific needs of processing document images. It shows that interdisciplinary collaboration can be beneficial in the context of machine learning, data processing and professionalization of image processing and retrieval systems.

The article which is discussed in this chapter has been published under the name “Large scale parallel document image processing” in Volume 6815 of the Society of Photo-Optical Instrumentation Engineers Conference Series in 2008.
3.1 Introduction

Document processing for the public requires vast amounts of data storage and computational power. Users are more attracted to a large data set where they can find a lot of interesting information than to small data sets via a lot of different interfaces. Professionalization is required on the side of the researchers working on the background technology and on the interface toward the users.

Opening the cultural heritage for the public is the task of our research group. This entails not only the scanning of millions of documents, but especially the creation of a search engine to make the Dutch cultural heritage accessible. The focus is on both the quality of the results of queries and the professionalization of the way the data is stored and queried by a large number of people.

*Scaling up* In the Netherlands, with a population of 16 million people, there are more than 150 thousand persons researching their genealogical background as a hobby. Outside the Netherlands people could also be interested because some of their ancestors were Dutch emigrants. Australia, the United States of America, Canada and the Dutch Antilles harbor millions of people who have Dutch ancestors. If all those users would use the final system once or twice a week with, let’s say, ten queries it means that the system has to process $\pm 3 \times 10^5$ queries a day. This is not feasible on a stand-alone PC, but still it has to be ready before the end of 2008. Instead of focusing only on the content of the pattern recognition (LeCun et al., 1998; Lambert Schomaker, 2004) the research at our department is focused as well on the scale of the document image processing pipeline. This article is mainly concerned with the scaling up of the technology, instead of the quality of the pattern recognition.

Experience has shown that a system of this scale requires better equipment than what is found in ordinary PC’s. The normal file system found in Linux and Windows is too slow. It requires a distributed data base system to handle the huge amounts of data together with distributed calculations.
Although the introduction to this chapter might suggest as if a negative conclusion is the most likely outcome, it is interesting to see that there are research groups from a domain other than AI who are working on generating massive amounts of computational power and data storage, in this case astrophysics. The system described in this article operates on a high-performance computing cluster and is stable in the sense that it does not crash, which is not always the case with new types of hard- and software. It is important to have the opportunity to scale the computations. If the complexity and amount of computations cannot grow in a dynamic but stable manner, GAI cannot be applied.

Interdisciplinary research The system that is implemented is based on a methodology borrowed from the astrophysics community, where there is a lot of experience in the processing of large amounts of data. The Astro-Wise system from the Kapteyn Institute is applied and adjusted to the problem of handwriting recognition: creating a scalable and searchable archive of historical handwritten documents, ready to be used by many users. The basic infrastructure is ready and uses the power of hundreds, up to thousands, of processors for research purposes. This opens up possibilities and new challenges for machine learning and artificial intelligence. Instead of toy problems, real-world and real-size problems are dealt with. The system that is in place now can compute in less than a day what would previously take more than a half a year on a single personal computer.

These increments in the quantity of data processing can be used to push the system out of its equilibrium. At the moment the system is not programmed in that manner, but the possibilities exist. Since systems out of equilibrium can demonstrate interesting properties of self-organization, it is good to know that the possibilities are available. Even if the computing and storage power of these massive computers are not enough to show interesting properties, it probably will be possible to do so in the (near) future by means of non-linear software components.

A daunting task: The focus of this paper is the setting up of a very large retrieval system implementing solutions of handwriting recognition. The problems described in this paper combine handwriting technologies and high performance computing. Often the computing part is set aside as being not very interesting. With this article the authors hope to convey that setting up a system that can deal with millions of high
resolution documents is a daunting and scientific task. Many computing problems have no standard solutions. Solutions that work well on one or a few computers probably do not scale up to hundreds or thousands of processors. If scientists want to get out of the world of toy problems, they have to address the issues concerning high performance computing. In some sense this article conveys our experiences while building this system. It is a mixture of several scientific areas such as supercomputing, human-computer interaction, handwriting recognition, patter recognition, image retrieval and machine learning.

The structure of this chapter: This chapter is structured as follows: In section 2 the collection from the national archive that is used is discussed, to give an idea of the type and amounts of data the Dutch government wants to make publicly available. Our department has several projects to create a search engine for these document images. The next section discusses the kind of problems encountered when dealing with connected cursive handwritten texts. Searching is very difficult since there is no one-to-one relation between the ink deposits and the accompanying Unicode. It also explains what the current research is and what is implemented on the cluster computer. Section 3 explains the system developed at the Kapteyn Institute in Groningen, which is augmented with dedicated machine learning and pattern recognition technology. Section 4 is about specific problems that one can encounter using massive parallel computations and the solutions from the high performance computing community used for the handwriting version of the Astro-Wise system. Section 5 puts it all together and the papers concludes with future directions.

3.2 The Dutch National Archive

The National Archive in the Netherlands is an institute concerned with the storage and preservation of documents. Most of them are old and many not in book form, but collections of sheets of handwritten pieces of paper such as letters of royals, bookkeepings of fourteenth century rulers and more. In total there is about one hundred kilometers of bookshelves filled with this material. The National Archive is specialized in state related documents such as the laws that have been discussed in the Dutch parliament, documents from the Dutch ministries and the Cabinet of the Queen. The Dutch name for this collection is “Het Kabinet der Koningin”, abbreviated as KdK.
The royal collection of the Cabinet of the Queen ensures the administrative support for the queen at the exercise of its constitutional tasks and acts as a link between queen and ministers. The prime minister carries the ministerial responsibility for the KdK.

The collection of the KdK comprises about 3 kilometers of boxes on shelves filled with (mostly) handwritten documents. Our group works with the index books of the KdK, which consists a few hundred thousand pages. On these pages there are references to documents with royal decrees. The content is highly structured and consists in total of approximately 25 million pages.

Creating a search engine for millions of documents and a data server of this magnitude is more difficult that what, for example, a Google or Yahoo is doing. Searching in images is more difficult than searching in Unicode. In the cultural heritage most of the queries are in the Unicode, but the scanned pages rarely have an annotation in Unicode space. The entire archive contains about eight hundred million pages of handwritten texts and the estimate is that in other archives in the Netherlands there are about seven times more documents. The creation of a search engine for this order of magnitude requires careful preparation.

At the moment there is no method to convert the document images reliably to Unicode. Current technologies are able to find similar pieces of ink. Whenever one of those pieces of ink is annotated by a human the machine estimates the probability that the other pieces of ink are close in Unicode space as well.

The biggest challenge of the search engine is what the users are interested in; They want to know who did what, and where did that person do it. The problem with this is that family names are exactly the type of data with the lowest frequency. It is much more difficult than finding common words with a high frequency, which are the ones that the users are the least interested in.

Could it be that the problem as described above is approached from the wrong perspective? The goal is to create a search engine for handwritten text, but perhaps there is a possibility to generate information out of the data, using some form of sorting mechanism, such as a human. If the data, perhaps information if it has passed the sorting mechanism, can self-organize to form meshworks then the system could be an implementation of an abstract search engine. The next chapter actually shows how a version of this abstract search engine is implemented in a complete system, but first the hardware has to be investigated further to see what is and what is not possible.
3.3 Connected cursive handwritten text

State of the art Optical Character Recognition technology is not sufficient for recognizing words in connected cursive handwriting. There is no obvious way to segment words into letters, since to recognize the word you have to recognize the letters, but to recognize the letters you need to know what the word is. It is often too difficult to segment the words since the ink is connected, or the gap between words is smaller than the gap between letters in words. Also letters with descenders, e.g. 'g', 'j', 'p', are often connected to the line of text below and the other way around for the ascenders, e.g. 't', 'k', 'l', 'b'. An example of the material is shown in figure 3.1.

Many problems with the quality of the processing have to be solved. For example: What is the best way to binarize an image? How can the computer identify a line of text? In boxed isolated-characters scripts (e.g. Japanese, Korean, Chinese) this is a lot easier than in western style scripts. Also between writers there is a lot of variation. Writers tend to write sloppy (think about the handwriting of a 'typical' physician) and the writing of a person can change during life.

Searching in connected cursive handwritten texts There are two types of search mechanisms often applied: Unicode search and Image Correlation. At the moment the Image Correlation research of \cite{Schomaker2007a} at our department is able to find parts of
images that are approximately the same. The advantage of this method is that it requires almost no knowledge of the language, but the disadvantage is that the upper bound of the complexity of calculations is $O(n^2)$, where $n$ is the amount of pieces of ink to compare. Even high performance computers are not fast enough to compare billions of little image parts with every other little image part.

Once correlated image parts are identified it is possible to query for the annotation (if available) and correlate them with each other. This method allows users to find a family of place name in the documents, creating a box around it and ask the system to give the pieces of ink that are similar.

Keyword-based searches gives us the text lines that are annotated and the pieces of ink that look like the results from the Unicode query. But annotation is expensive in time and money so most of the documents are not annotated. At the moment a mixed search system is in place which allows both of the systems to cooperate in assisting the users.

### 3.4 The Astro-Wise system

At the Kapteyn Institute the system is described as follows:

Astro-Wise stands for Astronomical Wide-field Imaging System for Europe. Astro-Wise is an environment consisting of hardware and software which has been developed to be able to scientifically exploit the ever increasing avalanche of data produced by scientific experiments. Astro-wise is a high capacity and longterm information system. As the name suggests, Astro-Wise started out as an system geared toward astronomy, but is now starting to be used also outside astronomy. Astro-Wise is an all-in-one system: it allows a scientist to archive raw data, calibrate data, perform post-calibration scientific analysis and archive all results in one environment. The system architecture links together all these commonly discrete steps in data analysis. The complete linking of all steps in data analysis, including the in- and output and used software code, for arbitrary data volumes has only been feasible thanks to a novel paradigm devised by the creators of Astro-Wise\(^1\).

The software is a combination of fast C/C++ routines that are connected via the

\(^1\)http://www.astro-wise.org
3.5 Parallel processing and high performance computing

Python programming language such as Python\(^1\). The system on the background is an Oracle database\(^2\) which is accessed through Python subroutines. The advantage is that people do not have to learn the SQL\(^3\) language and that many procedures are fully automated. For example: certain information or output can be made persistent, which means that it can not be deleted by normal users of the system. Only system administrators can do that. Another policy is to ensure that once a calculation is done it will not be done a second time, but retrieved instead.

Persistent information and remembering the procedures how an answer is generated from basic data elements is a form of the creation of a history in the computational system. A history is an essential element of GAI. The case presented here is probably not exactly what is required because almost *everything* is remembered, but that does not diminish the importance. One of the important elements of the Astro-wise system is that flows of calculations can be implemented (and why not generated?) to generate waves (Deleuze et al., 1987) of incremental knowledge buildup.

In python the procedures are defined that work on the data. For example: cut the pages are cut into line strips, which on their turn are correlated using the image correlation as mentioned above. The parameters of the cutting are made ‘persistent’ and so is the output. This way for every piece of data it is always possible to query the parameters and procedures used to create that piece of data.

Once modeled in Python/C it can be run on any amount of nodes of the cluster. The cluster divides the requests for calculations over the available nodes. On a node there is direct access to the database. The Astro-Wise system ensures a systematic journaling of experiments. It is always possible to see the outcome of any experiment. Professional large scale database systems provide convenient access mechanisms and have been well developed over the past decades.

3.5 *Parallel processing and high performance computing*

Working on more than one processor or core (from now on called node) requires some thinking. Working on a huge amount of nodes seems to be different than working on

\(^1\)http://www.python.org/
\(^2\)http://www.oracle.com/
\(^3\)http://en.wikipedia.org/wiki/sql
just a few. The solutions are geared toward using four hundred to twelve thousand nodes with the possibility to expand. The EGEE consortium\textsuperscript{1} for Enabling Grids for E-sciencE, the BOINC project (with as most famous instance the SETI@home project) and several Virtual Organizations are examples of groups that are slowly transforming the way people can use the Internet. This requires a different way of thinking about computation. The Internet is gearing toward a distributed storage and computation system in order to avoid bottlenecks and use the computational power of other groups and individual users who share their PCs at home.

At the moment we do not have a truly distributed system, in the sense that there is a central data storage server instead of a distributed one, but all the design choices keep the requirements in mind that it should not matter where the computations are being made or where the data is stored. This means that the ratio of computation versus data transfer should not be too small. The optimum is not known in advance but a rule of thumb is that the computational time should be at least between ten and a hundred time bigger than the time it takes to transfer the data to the node. Less would mean that the node is waiting too long. If the computing time takes too long then processors are waiting for each other too much and the risk of loosing a calculation becomes too big. This leads to another issue: the designer of the system has to be aware that any computation can get lost (computers crash, programs crash, local scheduling problems so it takes too long before the computation is ready). Therefore the nodes query the database to check not only which computation has to be done but also whether a computation was started long ago (for example 24 hours) and did not return any output. Such a life signal is often used in other complex projects, for example in robotics. If the node would still be functioning it would have updated the time stamp to let the other nodes know that it is still functioning appropriately.

*Embarrassingly parallel computing* There are a few ways to use high performance computing systems, all but one require complex methods for coordination. The simplest one is often referred to as: Embarrassingly parallel computing. The key feature is the independence of the calculations. If, for example, one wants to calculate the dynamics of a cubic kilometer of water, then every processor is depending on the calculations of its neighboring processors. In embarrassingly parallel systems there often are some dependences, but keeping it to a minimum eases the infrastructure of the

\textsuperscript{1}http://public.eu-egee.org
First some preprocessing is done and some analysis. Then the pages are cut into approximately 20 line strips and those line strips are stored on the data server. This process is tuned to last about ten minutes per calculation block. The data transfer itself takes about six seconds and the configuration of each node about ten. The ratio of computation versus data transfer is 40 to 1. The first step is to do this for all document images. Once there are many line strips, nodes that cannot get a document image for the preprocessing (since there are none) can start with the correlation of the $2 \times 10^4$ line strips, which amounts to $4 \times 10^8$ comparisons of line strips using a correlation on pixel level mechanism.

**Bottlenecks** Building the software of a high performance system consists of at least two phases. The first is the implementation of the software functionality and the second is getting rid of the bottlenecks that the hardware places on the software. The second phase usually consists of improving the software until one hits the hardware barriers. Bottlenecks can be caused by many processors requiring the same service or piece of soft- or hardware. Some of the more important bottlenecks that can happen are listed below, including the solutions we came up with.

*Data transfer from the server to the nodes* was too slow. On average it should have been more than enough, but all processors required data at the same time. The solution was to keep the processes on the nodes alive for much longer than what was required for the initial calculation and let the same process perform as many calculations as possible with the data stored locally. So the focus shifted from calculation-centered thinking to data-centered thinking. This does not solve the initial bottlenecks of all processors requiring data at the same time but it does solve the next bottlenecks of this type, because the times the nodes spend on computation had different lengths.

*Updating the database* requires consideration. It is imperative that the database does not have to many locks because too many nodes require its’ services. The solution is to have one process update the database with the outcomes of the computations from all the other nodes. This ensures that the data in the database is not corrupt and that it is singular. Even the best database systems have problems when, for example, twelve thousand processors require access to the same tables or service.

*Processes are not guaranteed to finish* since a node can break down on the hardware level and the software can crash unpredictably. A solution is to have a life signal that
Chapter 3. Mineralized substrate

Table 3.1: Performance increase per extra processor and in total

<table>
<thead>
<tr>
<th># processors</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>speedup</td>
<td>0.9999</td>
<td>0.9998</td>
<td>0.9876</td>
<td>0.4444</td>
</tr>
<tr>
<td>total (in years)</td>
<td>0.7610</td>
<td>0.0761</td>
<td>0.0071</td>
<td>0.0017</td>
</tr>
</tbody>
</table>

updates a field in the database once every few hours. Once a node starts processing a certain piece of data, which might take days, it updates a field every few hours with a time stamp. If any node needs new data to work on, it checks whether the data has been processed in total or not. If not, it checks whether the time stamp is older than twenty four hours. If so, then it is safe to start working on that data. The node updates the time stamp and all other nodes know, when querying the database for that data, that they should pick another piece of data.

Too many queries on the database creates locks. In the end the database works in a serial order. A process on a node therefore keeps track of the queries it made to minimize the queries on the database. It performs calculations on the data in memory instead of querying the database for every item. This also speeds up the process itself besides relieving the database.

Partitioning the tables in the database into smaller tables has the effect that for a query not the entire table has to be scanned, but only a portion of the table. In our case we have twenty thousand pieces of data that are correlated against each other. This creates a table with \((2 \times 10^4)^2 = 4 \times 10^8\) entries. With \(5 \times 10^5\) partitions this is a reduction from \(4 \times 10^8\) items to \((4 \times 10^8)/(5 \times 10^5) = 8 \times 10^4\) items per query that are processed and results in a substantial speedup.

The speedup could be, in the case of bottlenecks on the original one processor system, a bit more than linear with the amount of nodes. But often the high performance system creates its own bottlenecks and has less than linear increase in performance in the case of embarrassingly parallel computing. The actual speedup of any high performance system is inherently very difficult to measure (Zelkowitz et al., 2005). Since the Correlator (Schomaker, 2007a) has never been computed entirely for our data on a single PC it is not possible to give hard numbers. An estimate is based on the following reasoning: The time for a calculation is approximately 5 minutes for 1000 basic calculations/comparisons with the Correlator. It requires 10 seconds to set up a node and
node number 100 has to wait 10 minutes before it has all the data it needs. The increase in the waiting is linear. After the first data transfer it is not a significant bottleneck anymore. There are $4 \times 10^8 = 4 \times 10^5$ of basic calculation blocks of 600 seconds in total. The serial process would take, in total, $4 \times 10^5 \times 600$ seconds $\approx 7.6$ years. A node does, on average, $\frac{4}{#nodes} \times 10^5$ calculations of 600 seconds. The parallel process has an average bottleneck of 10 seconds of initialization + the average waiting time at initialization. The speedup per node for a certain amount of nodes (#nodes) equals:

$$\frac{T(calc)}{T(calc) + T(init) \times #nodes + T(\text{av.wait}) \times #nodes}$$

where $T(calc)$ stands for the total calculation time, $T(init)$ for the initialization time of a node, $T(\text{av.wait})$ for the average waiting time ($\approx \frac{\text{maxwaitingtime}}{2}$). The total calculation time is:

$$\frac{T(calc)}{\text{speedup.per.node} \times #nodes}$$

Looking at table 3.1 and figure 3.2 it is obvious that there is a bottleneck in the system. In this case it is the waiting time for the initial data package. The solution to this problem can not be solved in software. It requires a distributed data storage server instead of a centralized one. It is likely that a broader bandwidth from the proposed servers is also needed otherwise that will become the next bottleneck.

### 3.6 Discussion and future work

To create a search engine of this magnitude requires huge data servers and massive amounts of computational power. At the moment we have implemented the first part of the search engine on a two hundred dual core computer cluster with a petabyte data server. The system is still in a test phase. Our algorithms are now parallelized and ready to use with any type of distributed system.

The next step is to hook the Blue Gene supercomputer\(^1\) to the data server to be able to use more than six thousand dual core processors for the computations. At that moment we have to reconsider the data storage system as can be seen in figure 3.2. It has to be

\(^1\)http://en.wikipedia.org/wiki/blue_gene
Figure 3.2: Estimated total calculation time
parallelized to decrease the bottleneck created at the initialization when all processors require their initial data.

This chapter demonstrates that it will not be easy to set up a system that uses the principles from GAI for the automated scaffolding of artificial intelligence. Another conclusion could be that our present day hardware is ready to be used in different ways than humans have been doing in the past. If the principles from embarrassingly parallel computing are being used, then it does not have to be too difficult to use GAI in a research program. Even if it is difficult now, the trends are that the usage of these hard- and software infrastructures will become easier over time, allowing more focus on GAI, and less on the hardware processes.