# Combining Computational Algebra Systems

**Steve Linton** reporting work of many people, especially Alexander Konovalov, Dan Roozemund, Peter Horn

University of St Andrews

Third de Brun Workshop 2009

# Outline

- Motivations

# Outline

- Motivations
- History

# Outline

- Motivations
- History
- SCIEnce

# Outline

- Motivations
- History
- SCIEnce
- Future Directions

# Combining Capabilities

- The most traditional reason

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results
  - GAP and `nauty` in GRAPE for fast graph automorphisms

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results
  - GAP and `nauty` in GRAPE for fast graph automorphisms
  - GAP as a servce for symmetry-breaking in search

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results
  - GAP and `nauty` in GRAPE for fast graph automorphisms
  - GAP as a servce for symmetry-breaking in search
- Less work than adding capabilities to “home” system

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results
  - GAP and `nauty` in GRAPE for fast graph automorphisms
  - GAP as a servce for symmetry-breaking in search
- Less work than adding capabilities to “home” system
- and you continue to get the benefit of development in “foreign” system

# Combining Capabilities

- The most traditional reason
  - GAP and Maple in CHEVIE for handling generic character tables
  - Maple and the PVS theorem prover to get more reliable results
  - GAP and `nauty` in GRAPE for fast graph automorphisms
  - GAP as a servce for symmetry-breaking in search
- Less work than adding capabilities to “home” system
- and you continue to get the benefit of development in “foreign” system
  - unless they break the interface!

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version
- Some you didn't realise you wanted to use before you left home

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version
- Some you didn't realise you wanted to use before you left home
- Some may only be released as an online service

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version
- Some you didn't realise you wanted to use before you left home
- Some may only be released as an online service

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version
- Some you didn't realise you wanted to use before you left home
- Some may only be released as an online service

- Telnet, cut-and-paste and web browsers only get you so far

# Mixing Local and Remote

- Some mathematical software doesn't work on Windows
- Some requires large (and perhaps changing) databases
- Some is still under development and you want to use the latest version
- Some you didn't realise you wanted to use before you left home
- Some may only be released as an online service

- Telnet, cut-and-paste and web browsers only get you so far
- Would be nice to combine local and remote computations

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.
- Individual processors have almost stopped getting faster

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.
- Individual processors have almost stopped getting faster
- What they are getting is more numerous

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.
- Individual processors have almost stopped getting faster
- What they are getting is more numerous
- A modern gamers PC has 4 general purpose processors and maybe 240 specialized graphics processors

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.
- Individual processors have almost stopped getting faster
- What they are getting is more numerous
- A modern gamers PC has 4 general purpose processors and maybe 240 specialized graphics processors
  - Supercomputers with 100 000 plus cores already exist, millions of cores are planned.

# So That Things Will Keep on Getting Better

## 1990's Complexity Theory

All EXPTIME problems can be solved in linear time –Wait a linear number of years while Moore's law makes computers exponentially faster

- Unfortunately this has slowed down, if not stopped.
- Individual processors have almost stopped getting faster
- What they are getting is more numerous
- A modern gamers PC has 4 general purpose processors and maybe 240 specialized graphics processors
  - Supercomputers with 100 000 plus cores already exist, millions of cores are planned.
- If we want to solve bigger problems in five and ten years, we will need to exploit multiple processors

# The Stone Age

# The Stone Age

# The Stone Age

- 1990–99

# The Stone Age

- 1990–99
- GAP 3

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)
- GAP writes input files for other programme

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)
- GAP writes input files for other programme
- GAP invokes other program

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)

- GAP writes input files for other programme
- GAP invokes other program
- Program writes GAP input to a file, possibly aided by a translator and exits

# The Stone Age

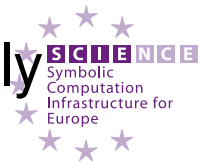
- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)

- GAP writes input files for other programme
- GAP invokes other program
- Program writes GAP input to a file, possibly aided by a translator and exits
- GAP reads and returns result.

# The Stone Age

- 1990–99
- GAP 3
- Packages interfacing GAP to specialized C stand-alone software
  - *p*-Quotient, nilpotent Quotient, Knuth-Bendix
  - Vector Enumerator, C meataxe
  - nauty (twice!)

- GAP writes input files for other programme
- GAP invokes other program
- Program writes GAP input to a file, possibly aided by a translator and exits
- GAP reads and returns result.
- Works OK – within fairly serious limitations



# The Bronze Age

# The Bronze Age

# The Bronze Age

- 1999–

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with
  - ACE coset enumerator, p-Quotient, nilpotent quotient, Knuth-Bendix

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with
  - ACE coset enumerator, p-Quotient, nilpotent quotient, Knuth-Bendix
  - KANT, Singular

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with
  - ACE coset enumerator, p-Quotient, nilpotent quotient, Knuth-Bendix
  - KANT, Singular
  - atlasrep package

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with
  - ACE coset enumerator, p-Quotient, nilpotent quotient, Knuth-Bendix
  - KANT, Singular
  - atlasrep package
- Works OK. Each interface is an individual challenge to program

# The Bronze Age

- 1999–
- GAP 4.2 and up supports interacting with other programs while they run
- String and file handling much improved to make it easier to program interactions
- Later XML parser, IO package (UNIX standard low level IO, including network)
- (Enhanced) packages for interacting with
  - ACE coset enumerator, p-Quotient, nilpotent quotient, Knuth-Bendix
  - KANT, Singular
  - atlasrep package
- Works OK. Each interface is an individual challenge to program
- SAGE is essentially built around this approach

# SCIEnce: Symbolic Computation Infrastructure for Europe


**SCIEnce** – **S**ymbolic **C**omputation **I**nfrastructure in **E**urope

http://www.symbolic-computation.org

5-year **Research Infrastructure** project supported by the EU Framework VI programme grant RII3-CT-2005-026133.

# Partner institutions

University of St Andrews, St Andrews, UK

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT
Universität Kassel, Germany

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT
Universität Kassel, Germany

TU/e
Technische Universiteit Eindhoven, Netherlands

SCIENCE Symbolic Computation Infrastructure for Europe

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT Universität Kassel, Germany

TU/e Technische Universiteit Eindhoven, Netherlands


Technische Universität Berlin, Germany

SCIENCE Symbolic Computation Infrastructure for Europe

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT
Universität Kassel, Germany

TU/e
Technische Universiteit Eindhoven, Netherlands


Technische Universität Berlin, Germany


Institute e-Austria Timisoara, Romania

SCIENCE Symbolic Computation Infrastructure for Europe

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT
Universität Kassel, Germany

TU/e
Technische Universiteit Eindhoven, Netherlands


Technische Universität Berlin, Germany


Institute e-Austria Timisoara, Romania

Maplesoft
Maplesoft, Waterloo, Canada

SCIENCE Symbolic Computation Infrastructure for Europe

# Partner institutions


University of St Andrews, St Andrews, UK


Research Institute for Symbolic Computation, Linz, Austria


Centre National de la Recherche Scientifique, France

UNIKASSEL VERSITÄT
Universität Kassel, Germany

TU/e
Technische Universiteit Eindhoven, Netherlands


Technische Universität Berlin, Germany


Institute e-Austria Timisoara, Romania

Maplesoft
Maplesoft, Waterloo, Canada


Heriot Watt University, Edinburgh, UK

SCIENCE Symbolic Computation Infrastructure for Europe

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services
  - simplify this OpenMath object

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services
  - simplify this OpenMath object
  - execute this string

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services
  - simplify this OpenMath object
  - execute this string
- or specialised

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services
  - simplify this OpenMath object
  - execute this string
- or specialised
  - look this up in your database

# Software Composability – one of the SCIEnce Activities

## Vision

Easy, robust and reliable way for users to create and consume services implemented in any of our (or other) systems.

- generic services
  - simplify this OpenMath object
  - execute this string
- or specialised
  - look this up in your database
  - do the step of my distributed computation specified by these parameters

SCIENCE Symbolic Computation Infrastructure for Europe

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)
- Robust client and server interfaces

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)
- Robust client and server interfaces
  - GAP, KANT, Maple, MuPAD, Macaulay 2, Magma (server only), Encyclopedia of Integer Sequences (server), . . .

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)
- Robust client and server interfaces
  - GAP, KANT, Maple, MuPAD, Macaulay 2, Magma (server only), Encyclopedia of Integer Sequences (server), . . .
  - Some need development versions, etc.

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)
- Robust client and server interfaces
  - GAP, KANT, Maple, MuPAD, Macaulay 2, Magma (server only), Encyclopedia of Integer Sequences (server), . . .
  - Some need development versions, etc.
- Java, C and C++ libraries making it easy to interface more systems

# What have we Delivered

- SCSCP – Symbolic Computation Software Composability Protocol (*née* Protocol X)
  - Simple lightweight remote procedure call protocol,
  - Control is XML
  - Data is passed as OpenMath (deep or shallow)
- Robust client and server interfaces
  - GAP, KANT, Maple, MuPAD, Macaulay 2, Magma (server only), Encyclopedia of Integer Sequences (server), . . .
  - Some need development versions, etc.
- Java, C and C++ libraries making it easy to interface more systems
- Supporting tools

# Three Applications

- Chosen mainly to try and convince you that it's easy to set up and use

# The application

## Some Nasty Polynomials

Let $p_1, p_2, \ldots, p_n$ be distinct prime integers. Define

$$P_{p_1,\ldots,p_n}(x) = \prod (x \pm \sqrt{p_1} \pm \sqrt{p_2} \pm \cdots \pm \sqrt{p_n})$$

# The application

## Some Nasty Polynomials

Let $p_1, p_2, \ldots, p_n$ be distinct prime integers. Define

$$P_{p_1,\ldots,p_n}(x) = \prod (x \pm \sqrt{p_1} \pm \sqrt{p_2} \pm \cdots \pm \sqrt{p_n})$$

- $P_{p_1,\ldots,p_n}$ has degree $2^n$ and is irreducible over $\mathbb{Z}$ but is highly reducible over every $\mathbb{F}_p$.

# The application

## Some Nasty Polynomials

Let $p_1, p_2, \ldots, p_n$ be distinct prime integers. Define

$$P_{p_1,\ldots,p_n}(x) = \prod (x \pm \sqrt{p_1} \pm \sqrt{p_2} \pm \cdots \pm \sqrt{p_n})$$

- $P_{p_1,\ldots,p_n}$ has degree $2^n$ and is irreducible over $\mathbb{Z}$ but is highly reducible over every $\mathbb{F}_p$.
- these polynomials, and others related to them are worst-case examples for standard factorisation methods.

# Setup in MUPad

```
>> package("OpenMath"):
>> swindyer:=proc(plist) <some details omitted> :
>> R := Dom::UnivariatePolynomial(x, Dom::Rational):
>> p1 := R(expand(swindyer([2,3,5,7,11]))):
>> p2 := R(expand(subs(swindyer([2,3,5,7,13,17])), x=3*x-2)):
>> p := p1 * p2:
>> degree(p), nterms(p)
96, 49
```

# Factoring

- In MuPAD

```
>> st := time(): F1 := factor(p): time()-st
38431
```

- in KANT, using a general-purpose KANT server that attempts to evaluate any OM objects sent to it

```
>> kant := SCSCP("scscp.math.tu-berlin.de", 26133):
>> st:=rtime(): F2:=kant::compute(hold(factor)(p)):rtime()
1221
```

- So calling the KANT service used only 1.2 seconds (wall-clock) instead of 38

# A Gröbner Basis Computation

## The Polynomials

$$(m_1 - a)^2 + m_2^2 - s^2, m_1^2 + (m_2 - b)^2 - s^2, (m_1 - a)^2 + (m_2 - b)^2 - s^2, -2ap_1 + 2bp_2, -2ap_2 - 2bp_1 + 4ab, aby - 1$$

# A Gröbner Basis Computation

## The Polynomials

$$(m_1 - a)^2 + m_2^2 - s^2, m_1^2 + (m_2 - b)^2 - s^2, (m_1 - a)^2 + (m_2 - b)^2 - s^2, -2ap_1 + 2bp_2, -2ap_2 - 2bp_1 + 4ab, aby - 1$$

- Computing a suitable GB of these polynomials is the main part of an automated proof of the circle theorem of Appolonius

# Setup in GAP

```
gap> R := PolynomialRing(Rationals,
>           ["a","b","s","y","m1","m2","p1","p2"]);;
gap> a := R.1;; b := R.2;; s := R.3;; y := R.4;;
gap> m1 := R.5;; m2 := R.6;; p1 := R.7;; p2 := R.8;;
gap> pols := [
>     (m1-a)^2 + m2^2 - s^2,
>     m1^2 + (m2-b)^2 - s^2,
>     (m1-a)^2 + (m2-b)^2 - s^2,
>     -2*a*p1+2*b*p2,
>     -2*a*p2-2*b*p1+2*a*2*b,
>     a*b*y-1
> ];;
```

- In GAP 4.4.10, computing this GB is hard

  ```
  gap> B := GroebnerBasis(pols, MonomialGrevlexOrdering());
  ```

  does not return in a reasonable time.

# Calling a server

- A Macaulay2 server is running in Eindhoven

```
gap> I := Ideal(R, pols);;
gap> B2 := EvaluateBySCSCP("Macaulay2-Groebner", [I],
>              "scscp.win.tue.nl", 26133);;
#I  Creating a socket ...
#I  Connecting to a remote socket via TCP/IP ...
#I  Got connection initiation message
#I  Request sent ...
#I  Waiting for reply ...
gap> B2.object;
[ b-2*m2, a-2*m1, -4*m2*p2+p1^2+p2^2, m1*p1-m2*p2,
  4*m1*m2-m1*p2-m2*p1, s^2-m1^2-m2^2, y*m1*p2+y*m2*p1-1,
  4*y*m2^2*p2-p1, 4*y*m1^2*p2-4*m1+p1 ]
```

- This takes only a few seconds, mainly in the wrapper.

# Identifying Groups of Order 512

- GAP contains a database of all groups of order up to 2000, except those of order 1024

# Identifying Groups of Order 512

- GAP contains a database of all groups of order up to 2000, except those of order 1024
- For all orders in the database not divisible by 512, groups can be “looked up” to find the unique isomorphic database group

# Identifying Groups of Order 512

- GAP contains a database of all groups of order up to 2000, except those of order 1024
- For all orders in the database not divisible by 512, groups can be “looked up” to find the unique isomorphic database group
- One of GAP’s extension packages `ANUPQ` provides this facility for groups of order 512

# Identifying Groups of Order 512

- GAP contains a database of all groups of order up to 2000, except those of order 1024
- For all orders in the database not divisible by 512, groups can be “looked up” to find the unique isomorphic database group
- One of GAP’s extension packages `ANUPQ` provides this facility for groups of order 512
- But `ANUPQ` needs a special binary compiled and is not available for Windows, so we might wish to make this feature available as a service to call from our GAP sessions on Windows clients.

# Service design

- How to encode the groups?

# Service design

- How to encode the groups?
  - OM is not very good at encoding groups. In particular, there is no suitable representation for PC groups.

# Service design

- How to encode the groups?
  - OM is not very good at encoding groups. In particular, there is no suitable representation for PC groups.
  - Since we're only expecting GAP clients, however, we can use a GAP-specific representation – the integer given by `CodePcGroup`.

# Service design

- How to encode the groups?
  - OM is not very good at encoding groups. In particular, there is no suitable representation for PC groups.
  - Since we're only expecting GAP clients, however, we can use a GAP-specific representation – the integer given by `CodePcGroup`.
- So our server will offer just one function `IdGroup512ByCode` which will take this number and return the position of the group in the standard list (also an integer).

# Server-side Setup

```
gap> LoadPackage("scscp");; LoadPackage("anupq");;
gap> IdGroup512ByCode := function( code )
>    local G, F, H;
>    G := PcGroupCode( code, 512 );
>    F := PqStandardPresentation( G );
>    H := PcGroupFpGroup( F );
>    return IdStandardPresented512Group( H );
>    end;;
gap> InstallSCSCPprocedure("IdGroup512", IdGroup512ByCode );
InstallSCSCPprocedure : procedure IdGroup512 installed.
gap> RunSCSCPserver("localhost",26133);
```

# Client-side Wrapper

```
gap> IdGroup512:=function( G )
>    local code, result;
>    if Size( G ) <> 512 then
>      Error( "|G|<>512\n" );
>    fi;
>    code := CodePcGroup( G );
>    result := EvaluateBySCSCP("IdGroup512ByCode", [ code ],
>                              "scscp.st-and.ac.uk", 26133);
>    return result.object;
>    end;;
```

# Client-side Operation

```
gap> IdGroup512( DihedralGroup( 512 ) );
[ 512, 2042 ]
```

# Popular Wisdom

If you have two processors, use them to solve two problems at once.

# Popular Wisdom

If you have two processors, use them to solve two problems at once.

- Good advice for most people, most of the time until recently

# Popular Wisdom

If you have two processors, use them to solve two problems at once.

- Good advice for most people, most of the time until recently
- Less tenable in the world of 4-core laptops and 24-core servers (next year)

# Popular Wisdom

If you have two processors, use them to solve two problems at once.

- Good advice for most people, most of the time until recently
- Less tenable in the world of 4-core laptops and 24-core servers (next year)
- If you want to keep solving bigger problems for the next ten years you will have to engage with parallel programming

# Popular Wisdom

If you have two processors, use them to solve two problems at once.

- Good advice for most people, most of the time until recently
- Less tenable in the world of 4-core laptops and 24-core servers (next year)
- If you want to keep solving bigger problems for the next ten years you will have to engage with parallel programming
  - Time for algorithm designers to start considering this.

# ParGAP

- Cooperman *et al* c. 2000.

# ParGAP

- Cooperman *et al* c. 2000.
- Added facilities to run multiple copies of GAP on different machines and pass messages between them

# ParGAP

- Cooperman *et al* c. 2000.
- Added facilities to run multiple copies of GAP on different machines and pass messages between them
- Various higher-level options for parallel programming

# ParGAP

- Cooperman *et al* c. 2000.
- Added facilities to run multiple copies of GAP on different machines and pass messages between them
- Various higher-level options for parallel programming
- Worked well "under ideal conditions" – not fault-tolerant

# ParGAP

- Cooperman *et al* c. 2000.
- Added facilities to run multiple copies of GAP on different machines and pass messages between them
- Various higher-level options for parallel programming
- Worked well "under ideal conditions" – not fault-tolerant
- Messages were sent in ASCII – inefficient and sometimes unreliable

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”
  - eg a server that acts as a front end to a bunch of other servers, distributing jobs among them

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”
  - eg a server that acts as a front end to a bunch of other servers, distributing jobs among them
- Used to verify the Modular Isomorphism Conjecture on groups of order 512 (2+ CPU years on UK national grid service).

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”
  - eg a server that acts as a front end to a bunch of other servers, distributing jobs among them
- Used to verify the Modular Isomorphism Conjecture on groups of order 512 (2+ CPU years on UK national grid service).
- 50% speedup multiplying lartge polynomials on three cores

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”
  - eg a server that acts as a front end to a bunch of other servers, distributing jobs among them
- Used to verify the Modular Isomorphism Conjecture on groups of order 512 (2+ CPU years on UK national grid service).
- 50% speedup multiplying lartge polynomials on three cores
- Reimplemented some of the ParGAP frameworks

# Parallel Computing with SCSCP

- Can start multiple SCSCP calls to different servers in parallel
  - Todd-Coxeter in one, KB in another
  - Splitting up a computation or search
- Can write “SCSCP middleware”
  - eg a server that acts as a front end to a bunch of other servers, distributing jobs among them
- Used to verify the Modular Isomorphism Conjecture on groups of order 512 (2+ CPU years on UK national grid service).
- 50% speedup multiplying lartge polynomials on three cores
- Reimplemented some of the ParGAP frameworks
- All still fairly experimental, but we’d welcome test applications

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt
- 2009–2013

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt
- 2009–2013
- Goal to support parallelism in GAP at all levels for users and programmers

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt
- 2009–2013
- Goal to support parallelism in GAP at all levels for users and programmers
  - Supporting use of multiple processors on a single computer within a single GAP session

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt
- 2009–2013
- Goal to support parallelism in GAP at all levels for users and programmers
  - Supporting use of multiple processors on a single computer within a single GAP session
  - Supporting distributed computing on clusters and supercomputers

# Future Plans – HPCGAP

- EPSRC project (Infrastructures programme): St Andrews, Edinburgh Parallel Computing Centre, Aberdeen, Heriot-Watt
- 2009–2013
- Goal to support parallelism in GAP at all levels for users and programmers
  - Supporting use of multiple processors on a single computer within a single GAP session
  - Supporting distributed computing on clusters and supercomputers
  - Taking the opportunity to do a lot of necessary reengineering

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP
  - Do all of these independent things

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP
  - Do all of these independent things
  - Try all of these until one of them works

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP
  - Do all of these independent things
  - Try all of these until one of them works
  - Do this in the background

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP
  - Do all of these independent things
  - Try all of these until one of them works
  - Do this in the background
  - Divide and conquer

# HPCGAP for Multicore

- Frameworks for easy parallel programming in GAP
  - Do all of these independent things
  - Try all of these until one of them works
  - Do this in the background
  - Divide and conquer
- Parallel versions of some library and kernel code – meataxe; random-Schreier; recognition? what else would you like?

# HPCGAP for Clusters

- Re-engineered message passing framework for clusters and supercomputers

# HPCGAP for Clusters

- Re-engineered message passing framework for clusters and supercomputers
  - ParGAP with binary data transfer and some higher level interfaces

# HPCGAP for Clusters

- Re-engineered message passing framework for clusters and supercomputers
  - ParGAP with binary data transfer and some higher level interfaces
  - Hopefully easier to install and manage

# HPCGAP for Clusters

- Re-engineered message passing framework for clusters and supercomputers
  - ParGAP with binary data transfer and some higher level interfaces
  - Hopefully easier to install and manage
- Even higher level frameworks on top of message passing

# HPCGAP for Clusters

- Re-engineered message passing framework for clusters and supercomputers
  - ParGAP with binary data transfer and some higher level interfaces
  - Hopefully easier to install and manage
- Even higher level frameworks on top of message passing
- Demonstrator applications

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing
  - We will develop some parallel algorithms

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing
  - We will develop some parallel algorithms
  - We'd welcome suggestions for what to prioritize

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing
  - We will develop some parallel algorithms
  - We'd welcome suggestions for what to prioritize
  - We will have tools in a year or two for other people to try their own parallel algorithms

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing
  - We will develop some parallel algorithms
  - We'd welcome suggestions for what to prioritize
  - We will have tools in a year or two for other people to try their own parallel algorithms
  - We're looking for challenge problems/applications

# Closing Remarks and Requests

- We have working tools for combining systems, and for coarse-grain parallel computing
  - We're looking for applications and feedback
- We're developing tools for finer grain and larger scale parallel computing
  - We will develop some parallel algorithms
  - We'd welcome suggestions for what to prioritize
  - We will have tools in a year or two for other people to try their own parallel algorithms
  - We're looking for challenge problems/applications
  - If you design interesting parallel algorithms, there should be a good way to implement them