KASH

Sebastian Pauli

Institut für Mathematik
Technische Universität Berlin

Overview

KANT Computational Algebraic Number Theory
(C-Library)

KASH KANT Shell
(GAP 3 with extensions)

QaoS Query algebraic objects System
(Database)

GiANT Graphical Algebraic Number Theory
(Graphical User Interface)

History

1987 KANT V1 (Fortran Library)

1992 KANT V2 (C Library), built on the Cayley platform

1994 KANT V4 (C Library), built on the Magma platform

1995 KASH 1.0 (KANT Shell), based on GAP 3

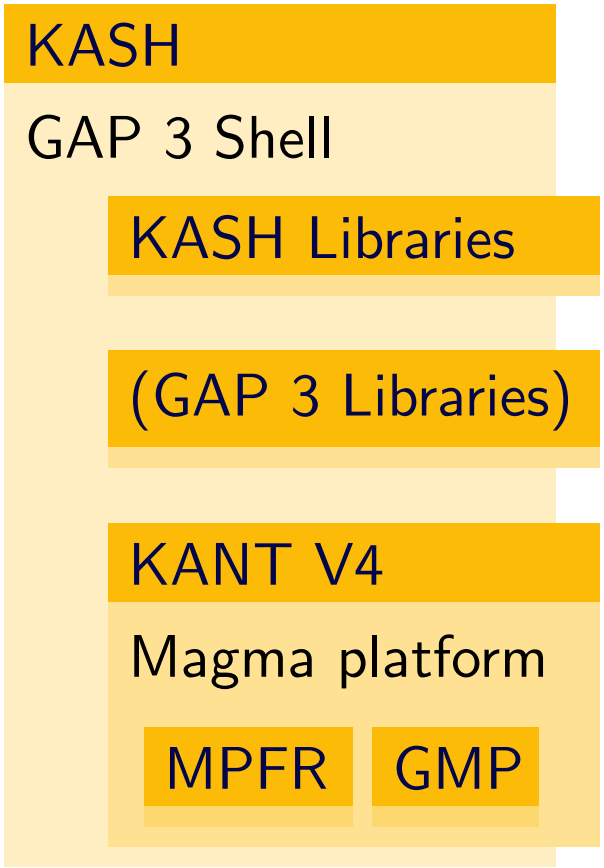
1996 KASH 1.7, Database for number fields

1999 KASH 2.1

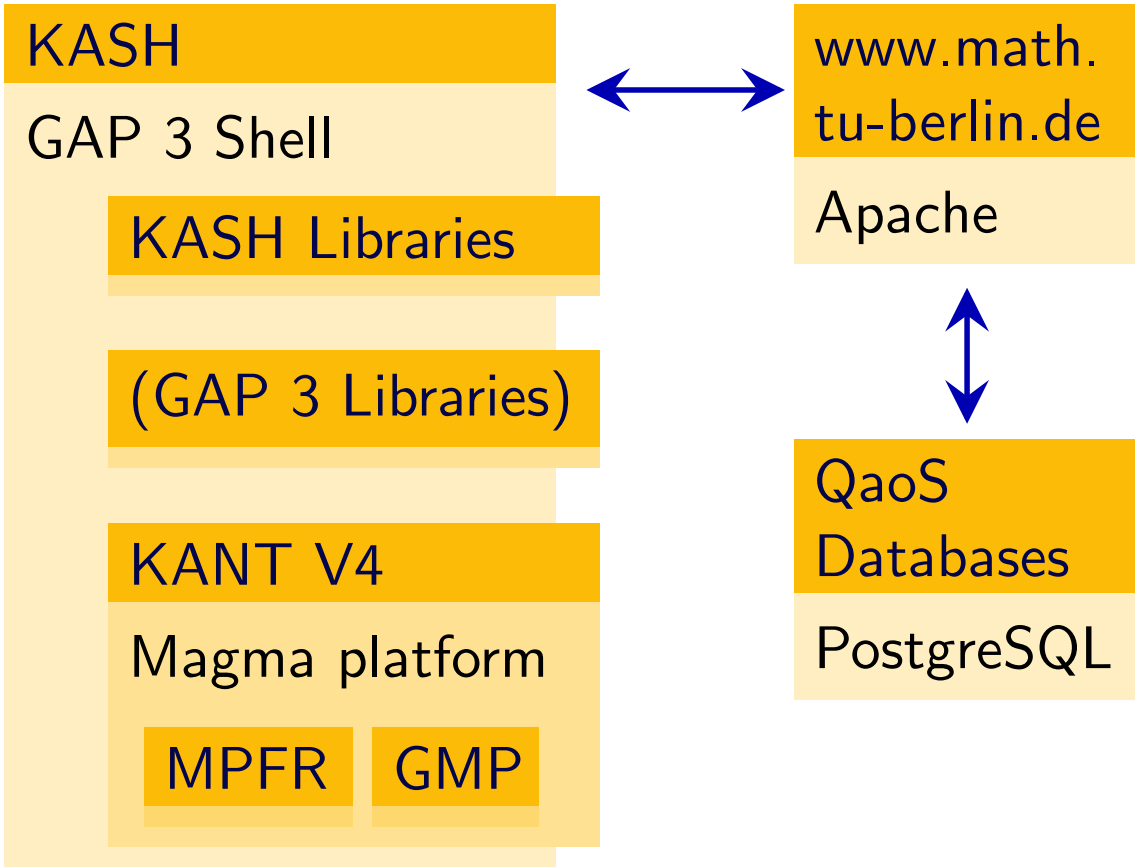
2004 KASH 2.4, WWW Database, webkash

2005 KASH 3, graphical user interface GiANT, QaoS Databases

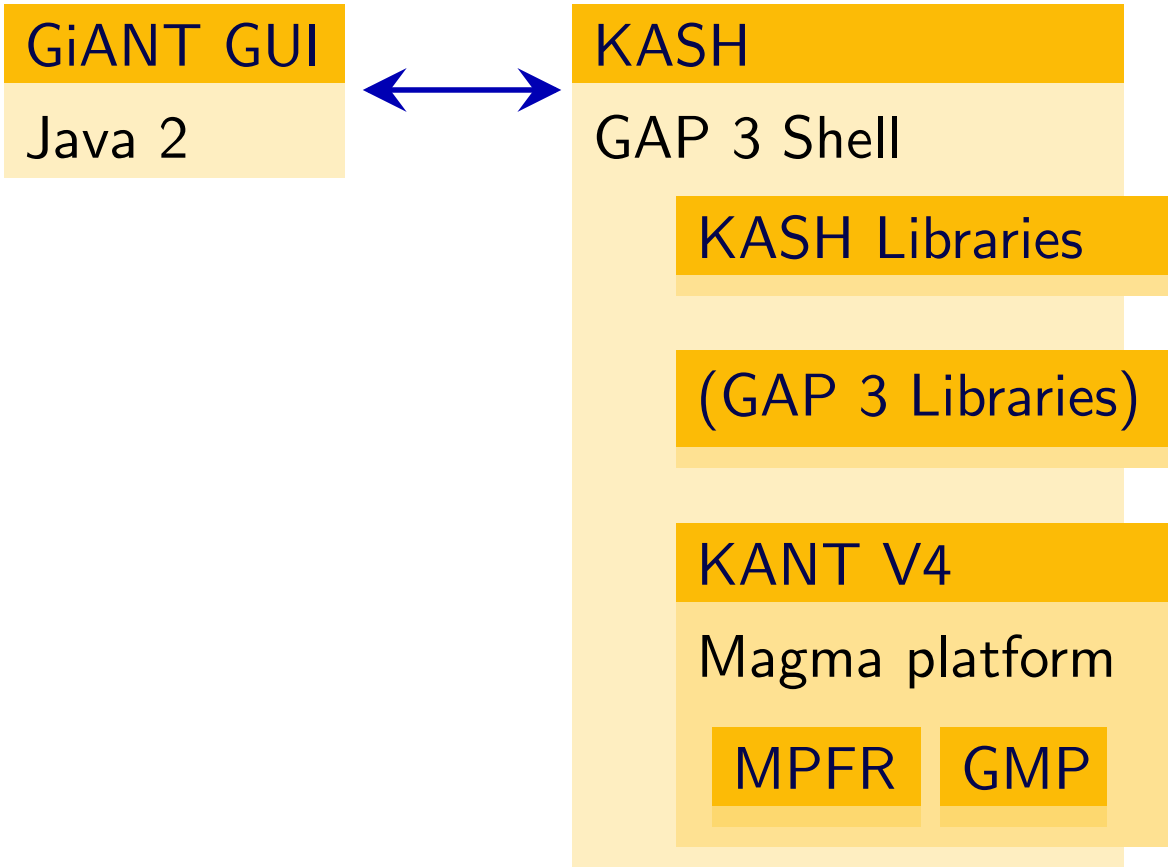
Architecture



Architecture



Architecture



KASH 3 – Features

Number Fields

- Integral Bases
- Galois Groups up to Degree 23
- Unit Groups
- Class Groups
- Class Fields

KASH 3 – Features

Number Fields

Function Fields

- Finite and Infinite Maximal Orders
- Unit Groups
- Riemann Roch Spaces
- Divisor Class Groups

KASH 3 – Features

Number Fields

Function Fields

Local Fields

- p-adic Fields
- Laurent Series
- Polynomial Factorization
- [Unit Groups]

KASH 3 – Features

Number Fields

Function Fields

Local Fields

Diophantine Equations

- Norm Equations
- Thue Equations
- Unit Equations

KASH 3 Help System

Features

- Inline documentation
- Represented as KASH records
- Help system written in the shell
- Handbooks and tutorials are compiled from documentation records
- HTML and print version via XML output
- Inline help with complex search patterns

KASH 3 Help System

Features

- Inline documentation
- Represented as KASH records
- Help system written in the shell
- Handbooks and tutorials are compiled from documentation records
- HTML and print version via XML output
- Inline help with complex search patterns

Search Patterns

- Full Text (?*)
- Name (?)
- Substring of Name (?!)
- Input Signature (?(*type*))
- Output Types (?->*type*)

Documentation Entries

Kinds of Documentation Entries

- Keyword
- Type
- Function
- Constant
- Statement
- Reference

Documentation Entries

Kinds of Documentation Entries

- Keyword
- Type
- Function
- Constant
- Statement
- Reference

Documentation Record Entries

- Name, Signature
- Examples
- Cross References
- Authors
- Literature References

KASH 3 Help Example

```
kash% ?!Type

1108: DecompositionType(<ord^num> 0, <elt-ord^rat> p) -> <seq()>
3243: Types
3277: NewType(<string> newtype, <elt-ord^rat> level) -> <type>
3279: ShowTypes()
3280: Type(<any> obj) -> <type>
3299: type

kash% ?3280
```

KASH 3 Help Example

```
kash% ?3280

#ab84be: FUNCTION (3280)
Type(<any> obj) -> <type>

PURPOSE:
Returns the type of 'obj'.  If 'obj' is extended, i.e., it is a
record with a 'base' component, then the type of the base component
is returned.  If 'obj' is a record with a 'type' component 'obj.type'
is returned.

EXAMPLE:
Type(3);

SEE ALSO:
3243: Types
3277: NewType(<string> newtype, <elt-ord^rat> level) -> <type>
3450: record
```

KASH 3 – Types

Goals and Constraints

- Matching of Datatypes from C-Library
- Representation of Categories
- Easy Type Matching for Overloading
- User Defined Types

KASH 3 Type System

Modular Types

$$[\texttt{elt}-]\,\langle \textit{algebraic structure}\rangle\,\hat{}\,\langle \textit{specifier}\rangle[\,/\,\langle \ldots \rangle]$$

KASH 3 Type System

Modular Types

$$[\texttt{elt}-]\,\langle \textit{algebraic structure}\rangle\texttt{\^{}}\langle \textit{specifier}\rangle[\,/\,\langle \ldots\rangle]$$

Examples

atoms for *algebraic structure*	`alg`, `fld`, `ord`, ...
atoms for *specifier*	`fun`, `num`, `pol`, ...

KASH 3 Type System

Modular Types

[elt–] ⟨*algebraic structure*⟩^⟨*specifier*⟩[/ ⟨...⟩]

Examples

atoms for *algebraic structure* `alg`, `fld`, `ord`, ...
atoms for *specifier* `fun`, `num`, `pol`, ...

`fld^fun`	Type of Function Fields
`ord^num`	Type of Orders of Number Fields
`elt-ord^num`	Type of Algebraic Integers
`alg^pol/fld^num`	Type of Polynomial Algebras over Number Fields

KASH 3 – Methods

Type Matching

`any`	matches any atom
`loc`	(local) matches `ser` (power series) or `pad` (p-adic)

KASH 3 – Methods

Type Matching

`any`	matches any atom
`loc`	(local) matches `ser` (power series) or `pad` (p-adic)

Examples

`elt-any^num`	matches the type of algebraic integers and numbers
`elt-ord^loc`	matches `elt-ord^pad`, `elt-ord^ser`

KASH 3 – Methods

Type Matching

`any`	matches any atom
`loc`	(local) matches `ser` (power series) or `pad` (p-adic)

Examples

`elt-any^num`	matches the type of algebraic integers and numbers
`elt-ord^loc`	matches `elt-ord^pad`, `elt-ord^ser`

Methods

`InstallMethod`(*documentation record*, *function*)

installs *function* as a method with name and signature given by the *documentation record*.

KASH 3 – Extended Objects

Records with `.base` Entry

Records are treated like their `.base` component by methods and internal functions.

KASH 3 – Extended Objects

Records with `.base` Entry

Records are treated like their `.base` component by methods and internal functions.

Other Special Record Entries

`.`*n*	*n*-th generator of a structure
`.type`	sets the type of a record
`.operations`	(as in GAP) for overloading of: `+`, `-`, `*`, `/`, `^`, `in`, `mod`, `Print`
`.ext`*n*	for multiple return values.

```
InstallMethod(
 rec(
  name := "Map",
  kind := "FUNCTION",
  sin  := [[any,"D"],[any,"C"],[func,"phi"]],
  sou  := [[map()]],
  short:= "Create a map with domain 'D' and codomain 'C' "
           +"from the function 'phi'.  ",
  see  := [DocHash("Composition(map(),map())")]
 ),
 function(domain,codomain,phi)
  return rec( base := phi,
      domain := domain,
      codomain := codomain,
      type := map(Type(domain),Type(codomain)),
      operations := rec(Print:= __Print_map, \*:=Composition)
      );
 end
);
```

```
InstallMethod(
 rec(
  name := "Composition",
  kind := "FUNCTION",
  sin  := [[map(),"phi"],[map(),"psi"]],
  sou  := [[map()]],
  short:= "The composition 'phi*psi' of the maps 'phi' and 'psi'.",
  ex   := ["add5 := Map(Z,Z,function(a) return a+5; end);"+
     "add5; add5(2); add10 := Composition(add5,add5); add10(2);"],
  see  := [DocHash("Map(any,any,func)")]
 ),
 function(phi,psi)
  return rec( base := function(a) return phi(psi(a)); end,
      domain := Domain(psi),
      codomain := Codomain(phi),
      type := map(Type(Domain(psi)),Type(Codomain(phi))),
      operations := rec(Print := __Print_map, \*:=Composition)
      );
 end
);
```

```
kash% add5 := Map(Z,Z,function(a) return a+5; end);
Mapping from ord^rat: Z to ord^rat: Z

kash% add5(2);
7

kash% add10 := add5*add5;
Mapping from ord^rat: Z to ord^rat: Z

kash% add10(2);
12
```

QaoS – Query algebraic objects System

by Sebastian Freundt, Robert Fraatz, P.

Database for Field Extensions

- Algebraic Extensions
- Transcendental Extensions

contains 1.3 million number fields

QaoS – Query algebraic objects System

by Sebastian Freundt, Robert Fraatz, P.

Database for Field Extensions

Database for Transitive Groups

contains all 40226 transitive groups of degree up to 30

QaoS – Query algebraic objects System

by Sebastian Freundt, Robert Fraatz, P.

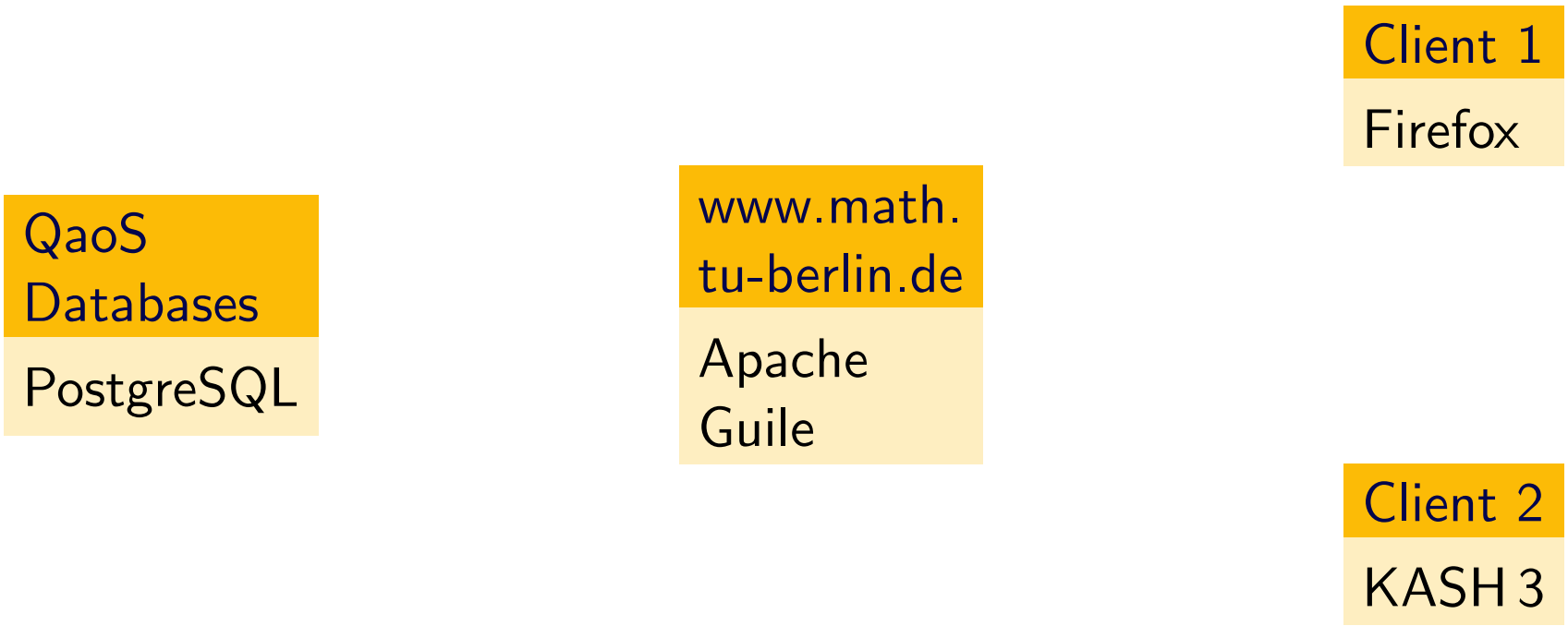
Database for Field Extensions

Database for Transitive Groups

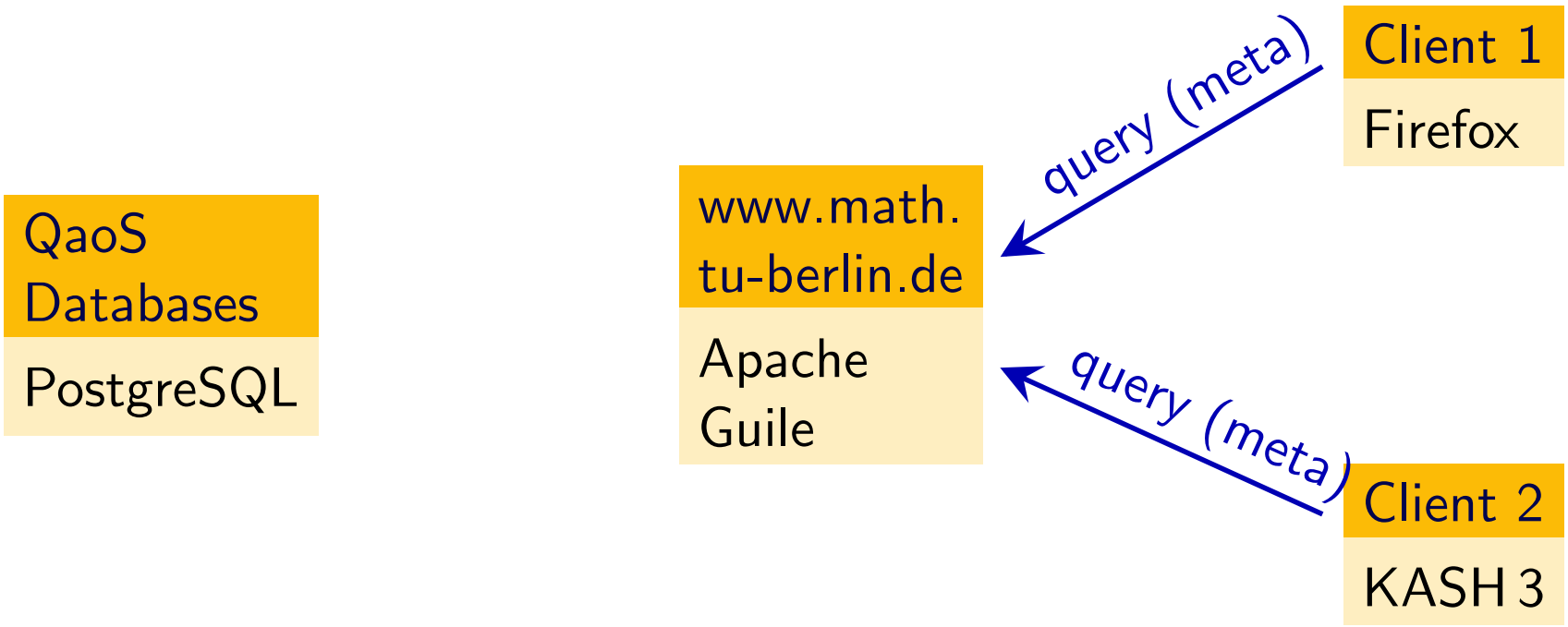
Access via HTTP from:

- GAP 4
- KASH 2.56
- KASH 3
- Maple
- webbrowser

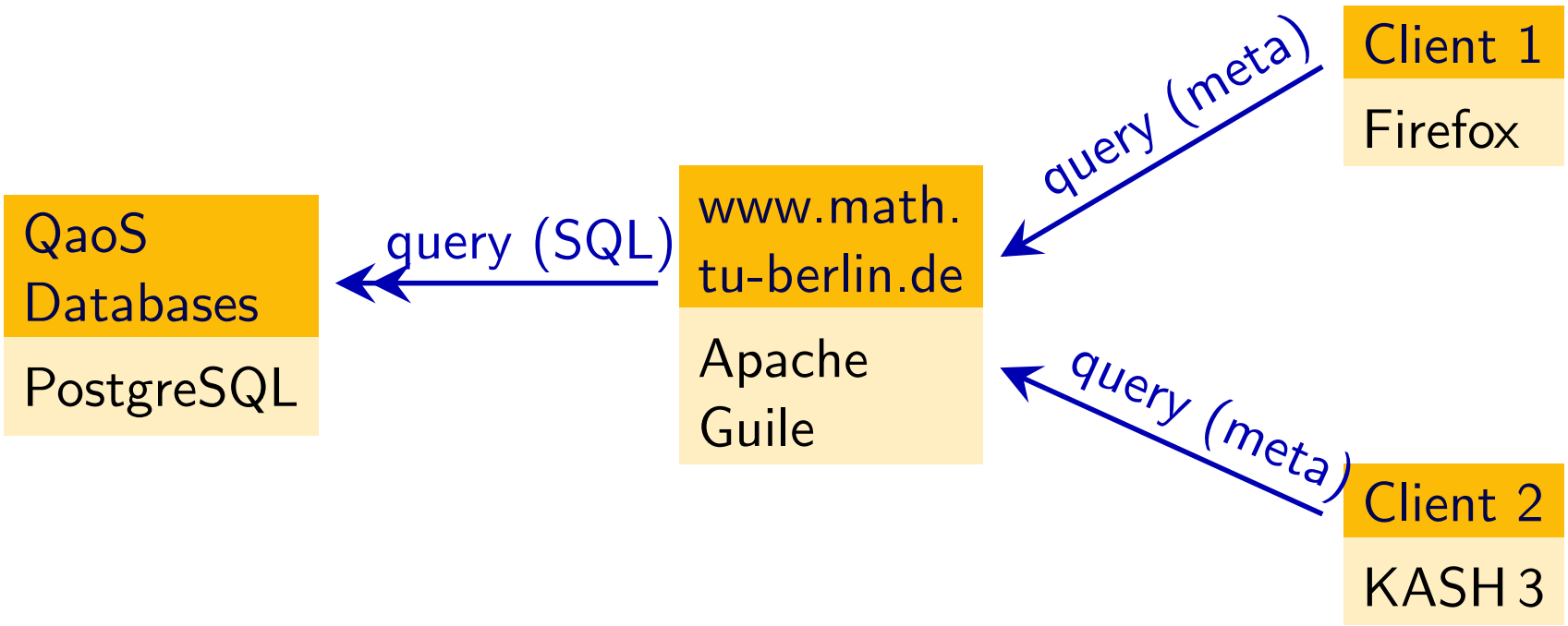
Using QaoS


Client 1
Firefox
QaoS
Databases
PostgreSQL
www.math.
tu-berlin.de
Apache
Guile
Client 2
KASH 3

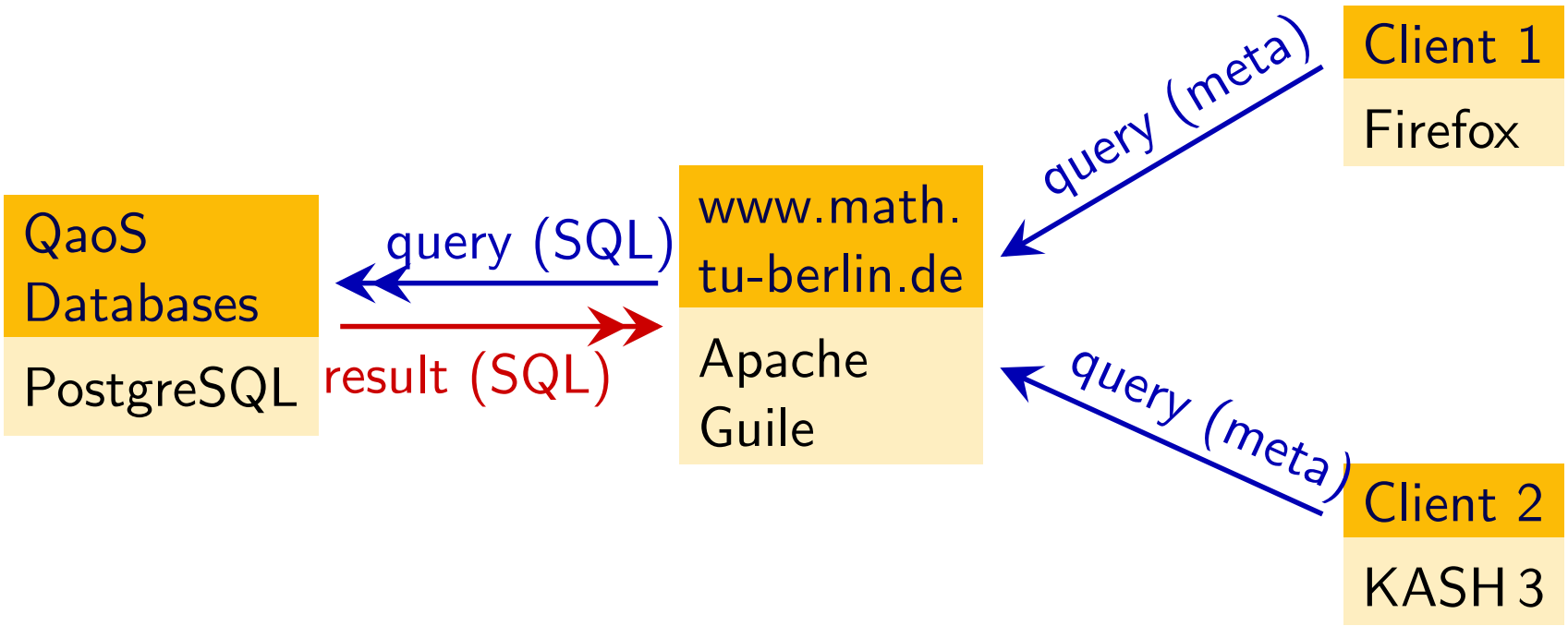
Using QaoS


QaoS
Databases
PostgreSQL
www.math.
tu-berlin.de
Apache
Guile
query (meta)
query (meta)
Client 1
Firefox
Client 2
KASH 3

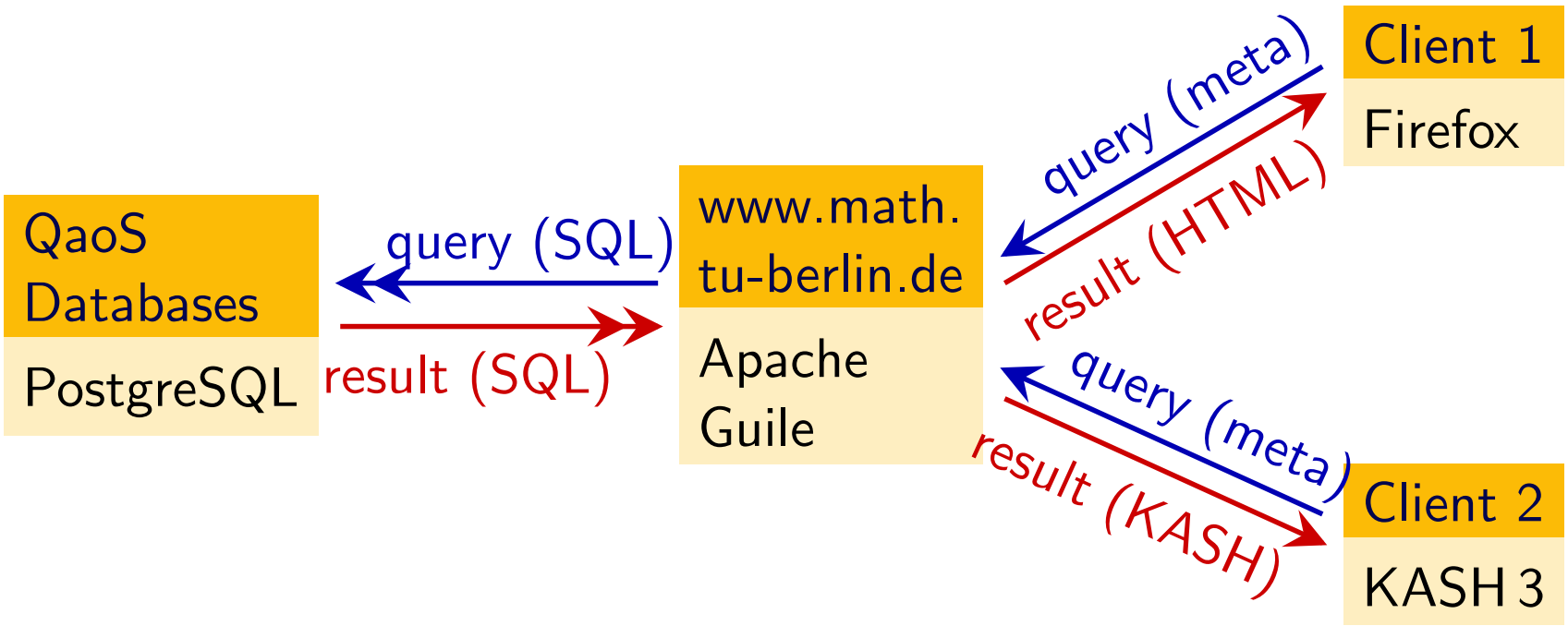
Using QaoS


Client 1
Firefox
query (meta)
www.math.
tu-berlin.de
Apache
Guile
query (SQL)
QaoS
Databases
PostgreSQL
query (meta)
Client 2
KASH 3

Using QaoS


Client 1
Firefox
query (meta)
www.math.
tu-berlin.de
Apache
Guile
query (SQL)
result (SQL)
QaoS
Databases
PostgreSQL
query (meta)
Client 2
KASH 3

Using QaoS


QaoS
Databases
PostgreSQL
query (SQL)
result (SQL)
www.math.
tu-berlin.de
Apache
Guile
query (meta)
result (HTML)
query (meta)
result (KASH)
Client 1
Firefox
Client 2
KASH 3

QaoS – Future Developments

Data

- Extension of tables of number fields
- Tables of function fields

QaoS – Future Developments

Data

- Extension of tables of number fields
- Tables of function fields

Access

- More clients
- OpenMath XML data format for transfer of data
- Bidirectional access will allow external users to add data

GiANT – Graphical Algorithmic Number Theory

written by Aneesh Karve

Evolution of Computer Algebra Systems

Libraries	$\longrightarrow$	Shells	$\longrightarrow$	Graphical User Interfaces
KANT	$\longrightarrow$	KASH	$\longrightarrow$	GiANT

GiANT is a Desktop Environment for working with Number Fields

GiANT – Graphical Algorithmic Number Theory

written by Aneesh Karve

Evolution of Computer Algebra Systems

Libraries	$\longrightarrow$	Shells	$\longrightarrow$	Graphical User Interfaces
KANT	$\longrightarrow$	KASH	$\longrightarrow$	GiANT

GiANT is a Desktop Environment for working with Number Fields

- Desktop shows Towers of Field Extensions
- Moving of Elements, Ideals, and Polynomials by Drag & Drop
- Arithmetic with Elements, Ideals, and Polynomials
- Computation of Invariants
- Architecture: KASH 2.5, Java 2

KASH can be downloaded from:

`http://www.math.tu-berlin.de/~kant`

GiANT is released under the GPL, see:

`http://giantsystem.sourceforge.net`

Thank You