Software architectures for modern data in the VO

Challenges and technics

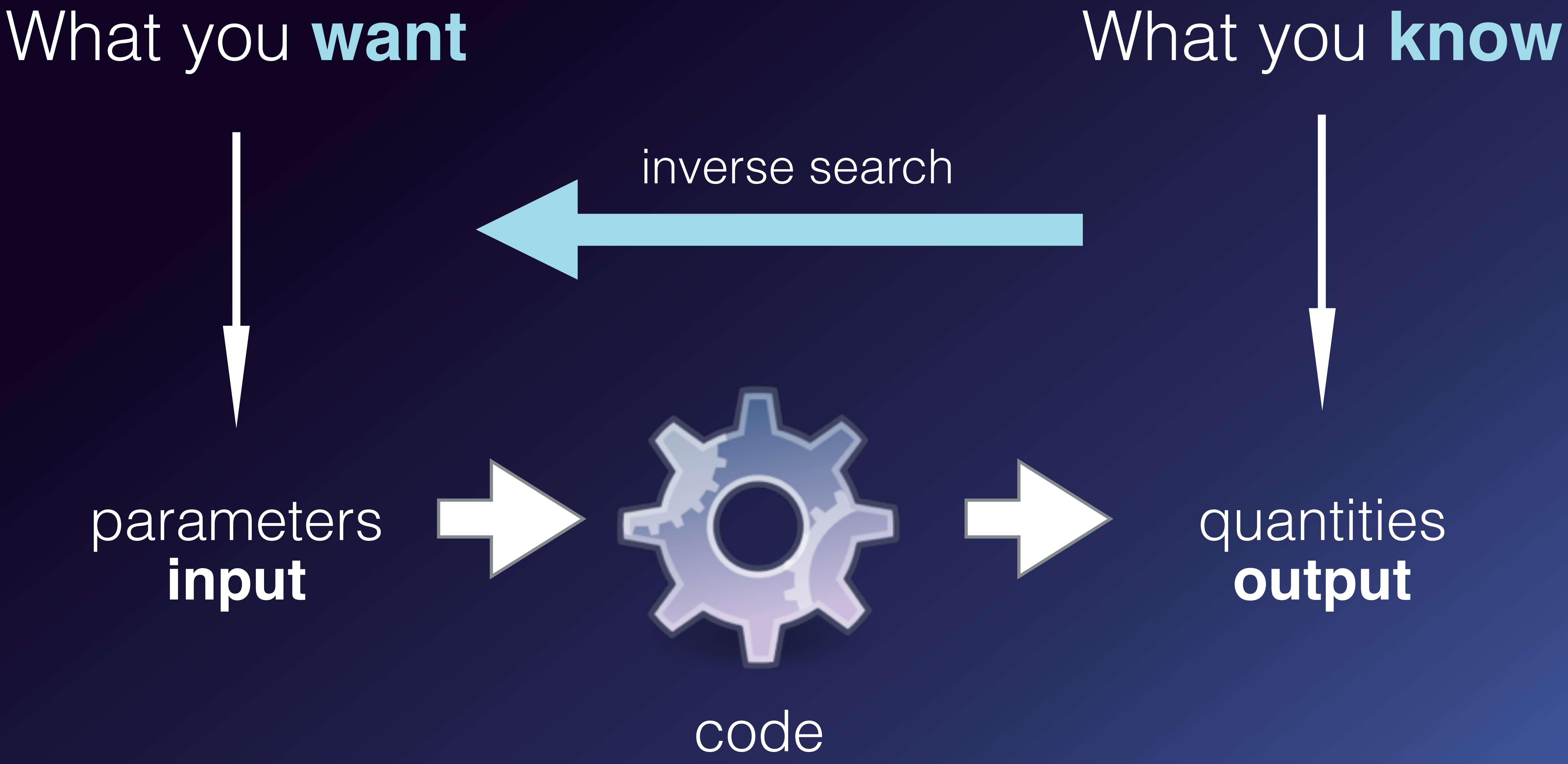
david.languignon@obspm.fr

2015.06.18 - Sesto - IVOA Interoperability meeting

Modern data challenges old software architectures

- Datasets are big, Big Data(set)
- **But not the only one problem**
    - datasets are chunks distributed in distant nodes
    - objects are very different & may have lots of properties

PDR Database



What you **want**

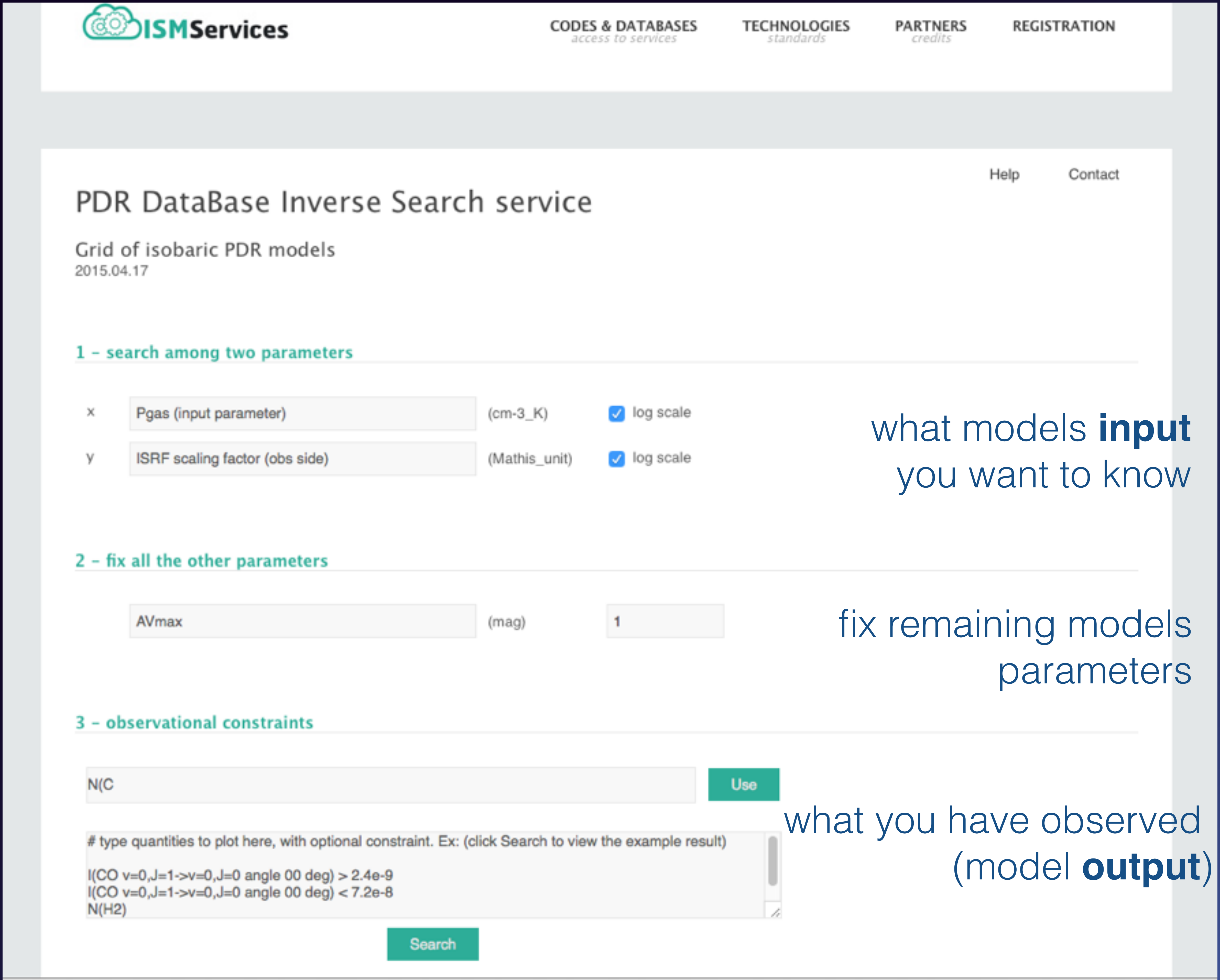
What you **know**

inverse search

parameters
input

code

quantities
output



ISMServices

CODES & DATABASES
access to services

TECHNOLOGIES
standards

PARTNERS
credits

REGISTRATION

Help Contact

PDR DataBase Inverse Search service

Grid of isobaric PDR models
2015.04.17

1 – search among two parameters

x	Pgas (input parameter)	(cm-3_K)	☑ log scale
y	ISRF scaling factor (obs side)	(Mathis_unit)	☑ log scale

what models **input**
you want to know

2 – fix all the other parameters

AVmax	(mag)	1

fix remaining models
parameters

3 – observational constraints

N(C

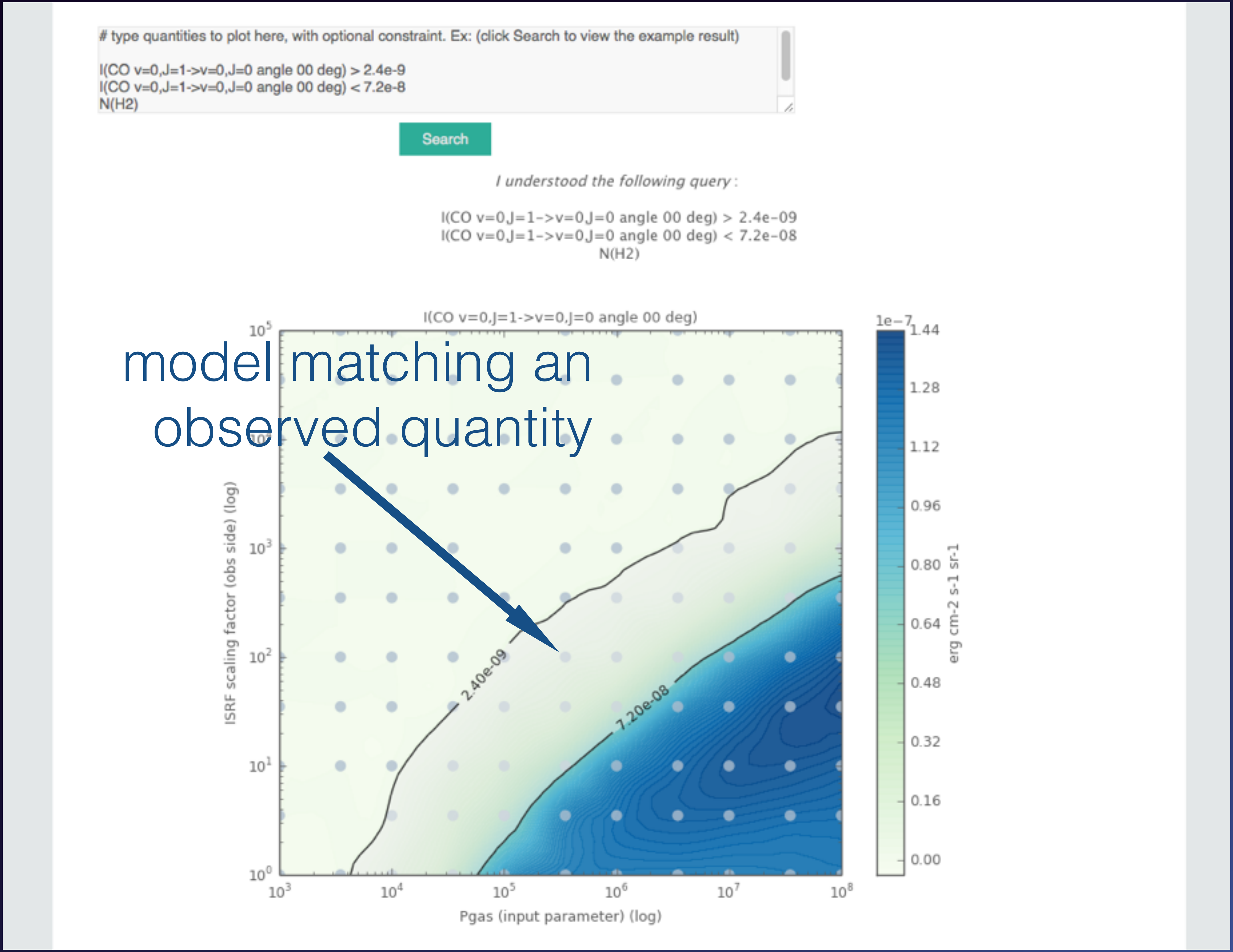
Use

```
# type quantities to plot here, with optional constraint. Ex: (click Search to view the example result)

I(CO v=0,J=1->v=0,J=0 angle 00 deg) > 2.4e-9
I(CO v=0,J=1->v=0,J=0 angle 00 deg) < 7.2e-8
N(H2)
```

Search

what you have observed
(model **output**)



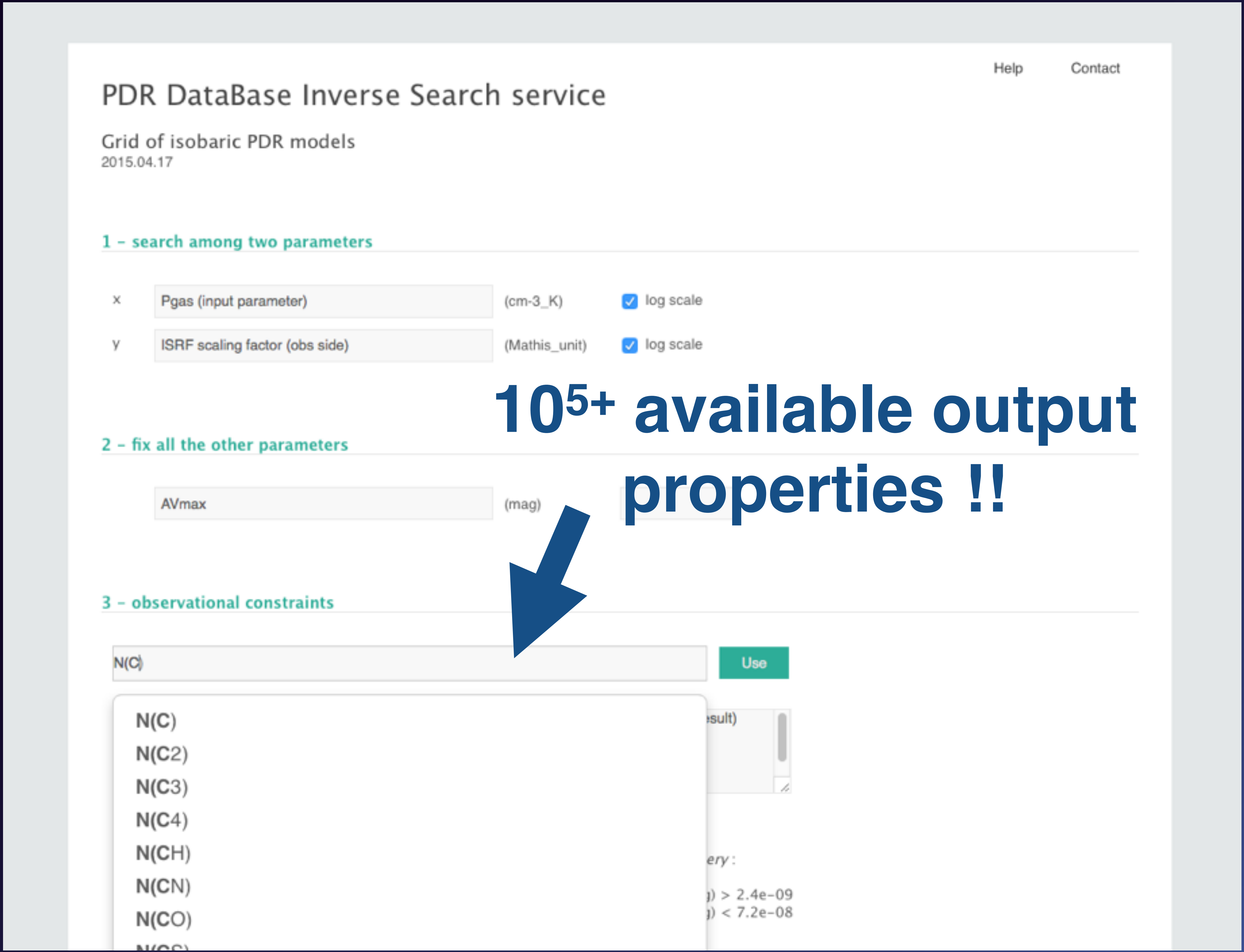
```
# type quantities to plot here, with optional constraint. Ex: (click Search to view the example result)

I(CO v=0,J=1->v=0,J=0 angle 00 deg) > 2.4e-9
I(CO v=0,J=1->v=0,J=0 angle 00 deg) < 7.2e-8
N(H2)
```

I understood the following query :

I(CO v=0,J=1->v=0,J=0 angle 00 deg) > 2.4e-09
I(CO v=0,J=1->v=0,J=0 angle 00 deg) < 7.2e-08
N(H2)

model matching an observed quantity



Help Contact

PDR DataBase Inverse Search service

Grid of isobaric PDR models

2015.04.17

1 – search among two parameters

x Pgas (input parameter) (cm-3_K) ☑ log scale

y ISRF scaling factor (obs side) (Mathis_unit) ☑ log scale

2 – fix all the other parameters

AVmax (mag)

3 – observational constraints

N(C) Use

N(C)
N(C2)
N(C3)
N(C4)
N(CH)
N(CN)
N(CO)

sult)

ery :

) > 2.4e-09
) < 7.2e-08

10^{5}+ available output properties !!

PDRDb inverse search

Use case

object	prop1	prop2	…	propN
o1	12	rad		1,2E+04
o2	…			
…				
oM				

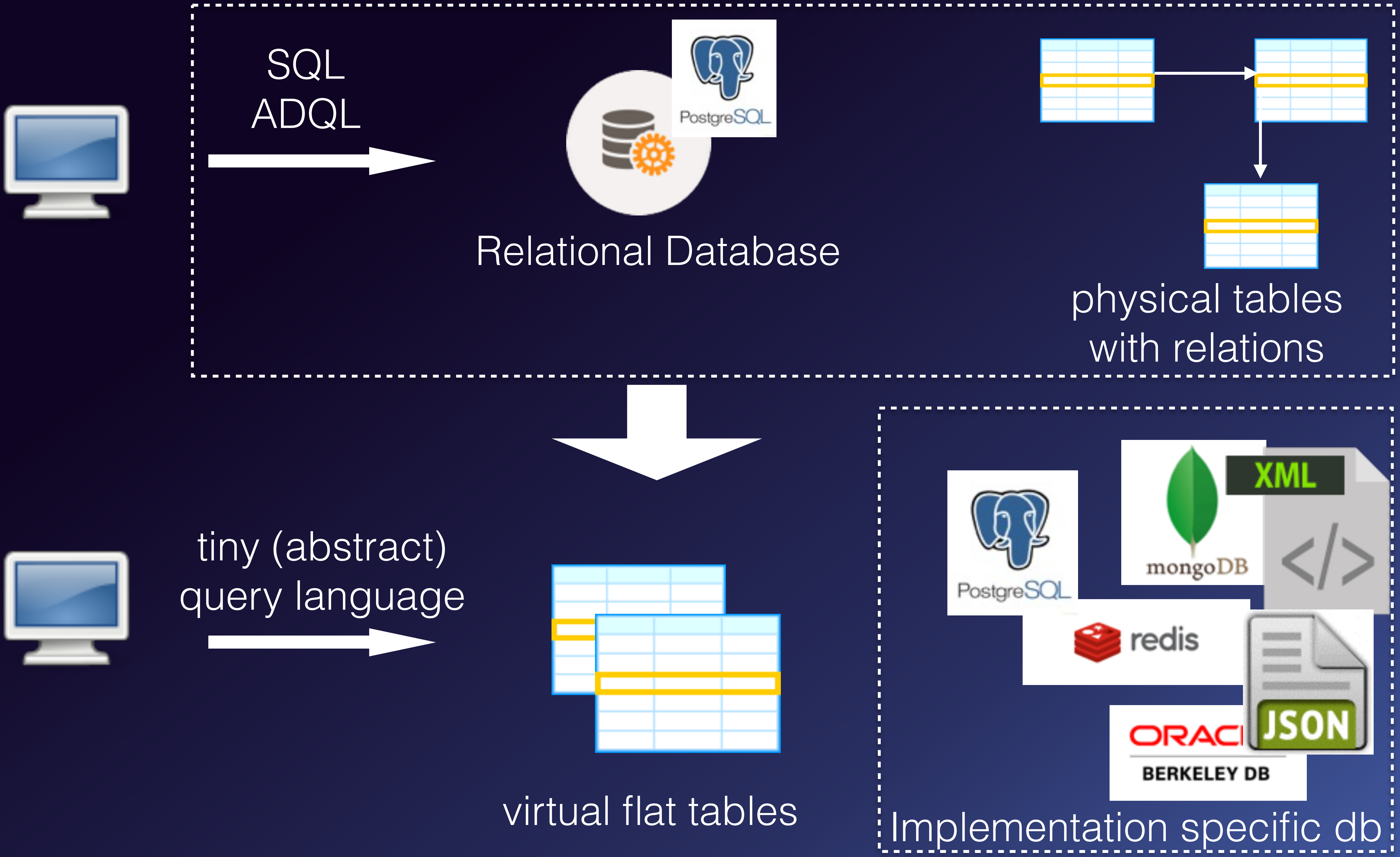
N is BIG

M is common

PDRDb inverse search

- We need flat tables with no relations but lots of columns
- We want to keep close to TAP semantics, TAP_SCHEMA
- But, get rid of relational dbs, with columns number limits
- Actually, get rid of any implementation tight coupling

PDRDb inverse search



SQL
ADQL

Relational Database

physical tables
with relations

tiny (abstract)
query language

virtual flat tables

Implementation specific db

VOTable as virtual table abstraction

- Map virtual flat tables to server-side VOTable
- Query through basic query language for flat tables
  - select, where
- Doing so, we have for free:
  - data exchange through VOTable xml serialisation
  - table schema (VOTable header)
  - VOTable have virtually infinite number of columns

Transparent sync/async

- You don't always know if it's best to choose either sync or async.
- The server may have more data than you to choose the best solution given a specific environment.
- Still allows manual settings for specific situations
  - I want async if available, do not auto-choose sync.

Transparent sync/async

- /resource

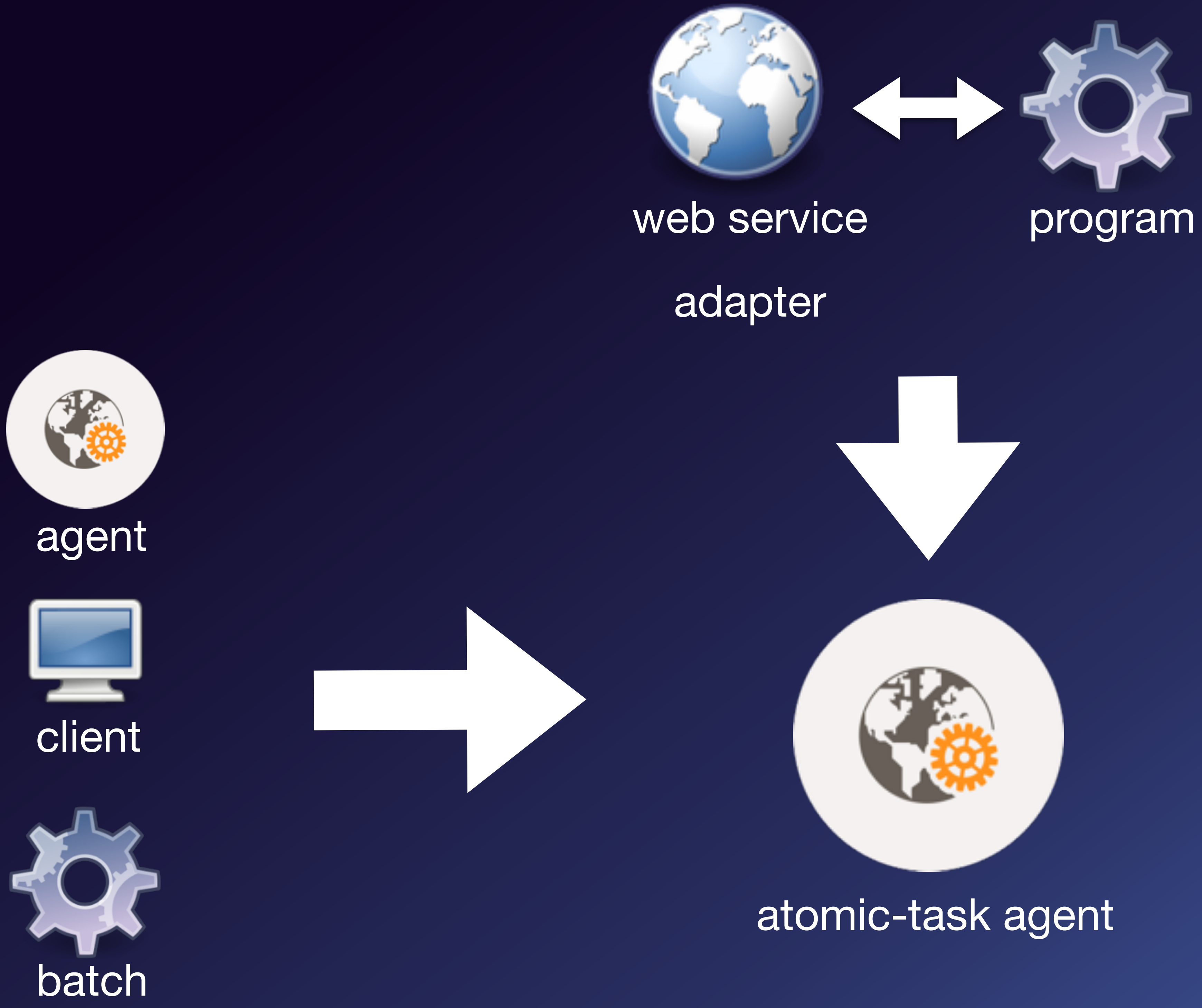
```
response = {
             ‘job_state’: ‘queued’,
             ‘sync’: ‘async’,
             ‘result’: <monitoring resource>
}


response = {
             ‘job_state’: ‘completed’,
             ‘sync’: ‘sync’,
             ‘result’: 23.3
}
```

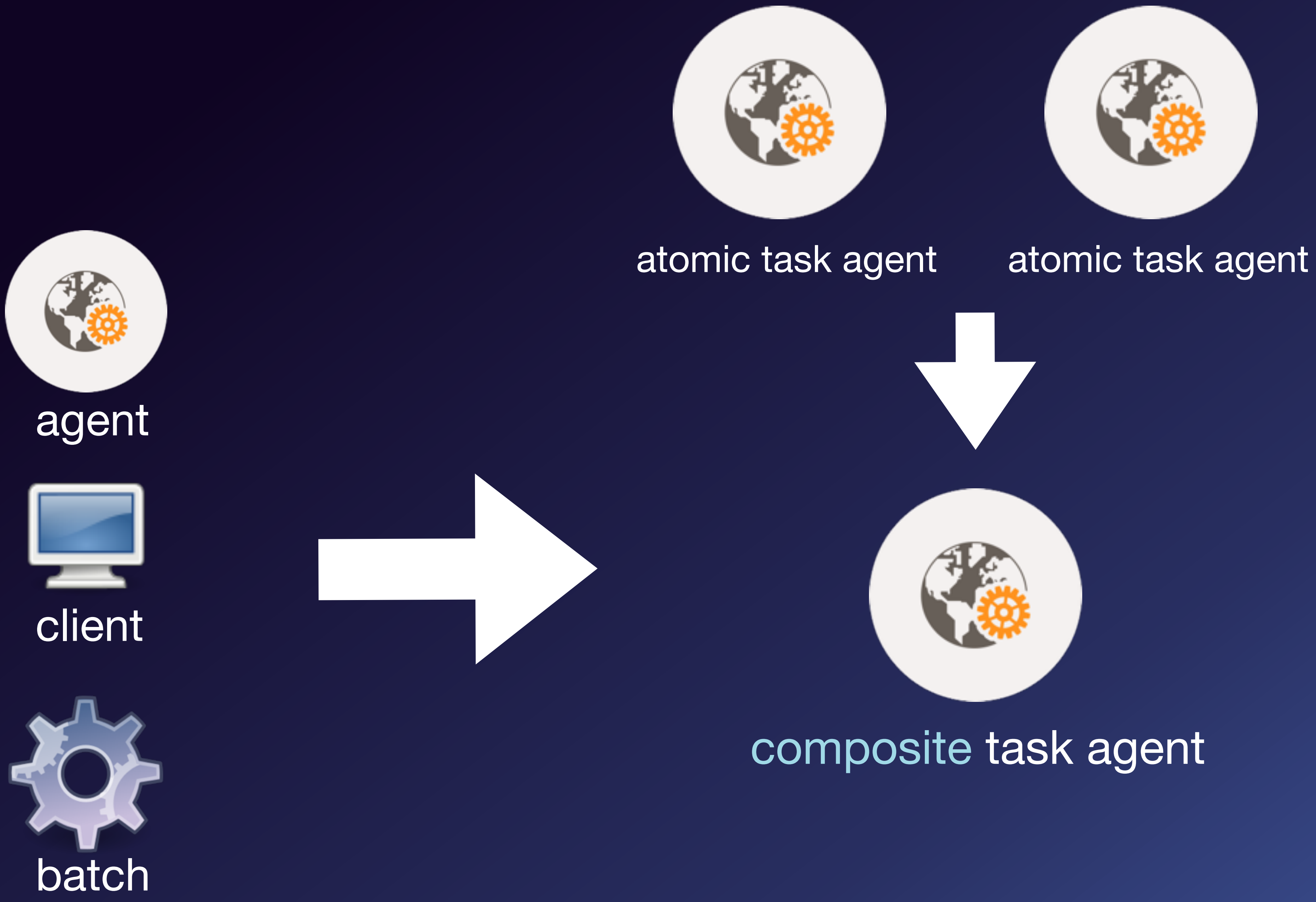
Distributed jobs communication and synchronisation

- Micro services, composite services, SOA
  - move the software close to the data
  - no longer the other way around !
- Channels as interprocess communication (CSP, Hoare, 1978)

Micro services architecture



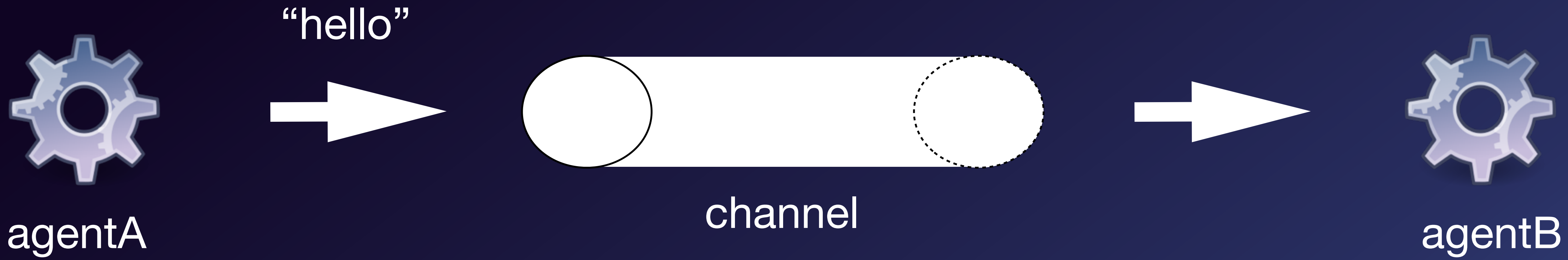
Micro services architecture



Micro services architecture

- Small
- Well defined functional perimeter
- Easy to debug & maintain
- Easy to document & delegate
- Easy to set up close to the data
- The complexity is shifted to services communication

Channels



Channels

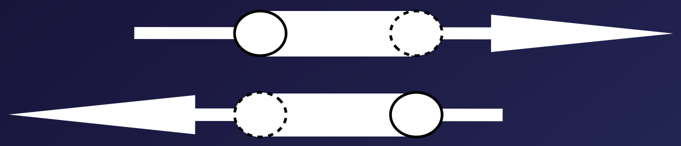
os process

main process

channel

thread

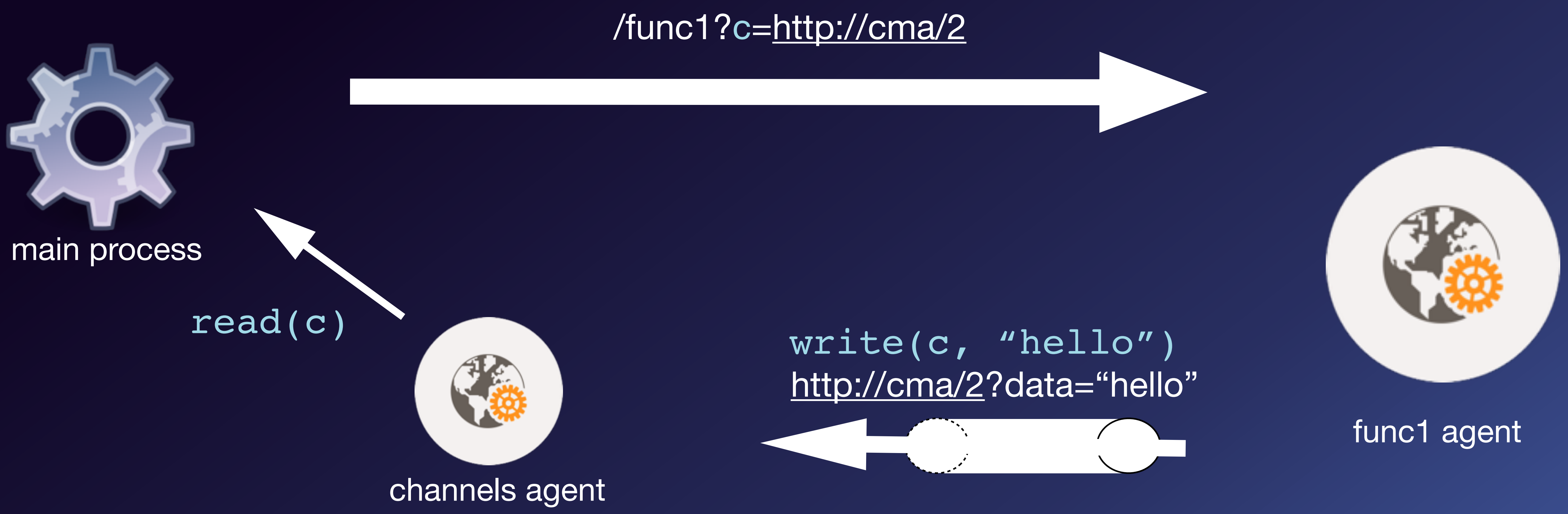
```
c := make_channel(string)
new_thread(func1, c)
print(read(c))
```



```
func1(c chan){
   write(c, “hello”)
}
```

Channels

distributed processes



/func1?c=http://cma/2

main process

`read(c)`

channels agent

`write(c, “hello”)`

http://cma/2?data=“hello”

func1 agent

Channels

- UWS 1.1 blocking alternative
  - /async?c=http://channel-resource
  - service write to channel resource (c), ex: JOBID, STATE
- Remove networks polling
- Alternative sync wrapping (cf uws 1.1 draft)

Channels

- But client must setup a channel resource
  - best suited for server-side client (batch process)
  - better for interoperability: oriented towards other services instead of Human user.
  - handle concurrent process very nicely (CSP/ blocking channel)
  - a protocol must be set between consumer / producer (uws 1.1 blocking is fine).

Case study
process the models of a grid

- How to configure all the jobs of a grid at once ?
    - compact parameter language (cpl): array slicing, list

job1

parameter	value
a	1
b	2.3
c	20

job2

parameter	value
a	2
b	2.3
c	20

job3

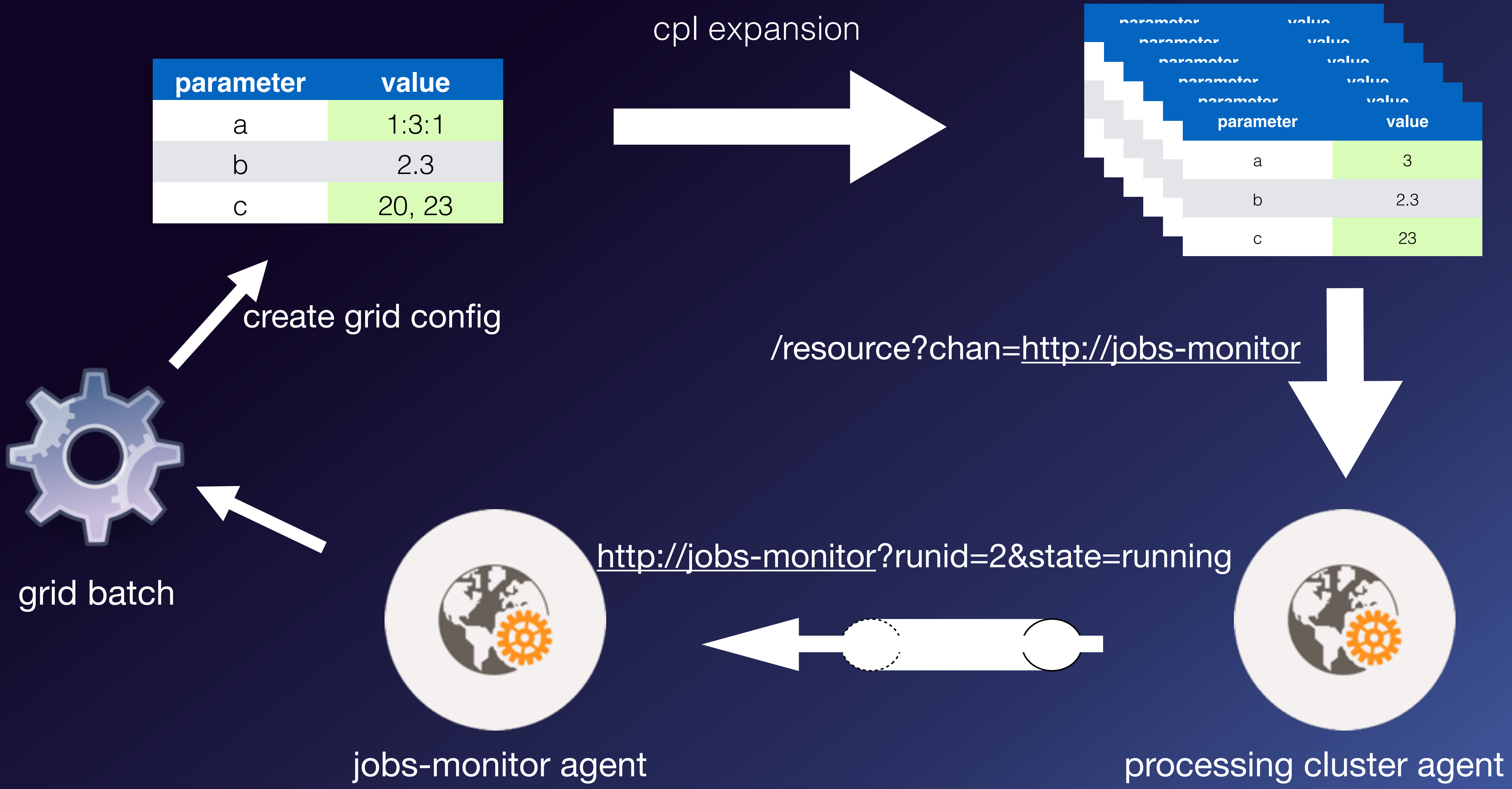
parameter	value
a	3
b	2.3
c	20

cpl v1

start:stop:step
val1, val2, …

parameter	value
a	1:3:1
b	2.3
c	20

Case study
process the models of a grid



parameter	value
a	1:3:1
b	2.3
c	20, 23

parameter	value
a	3
b	2.3
c	23

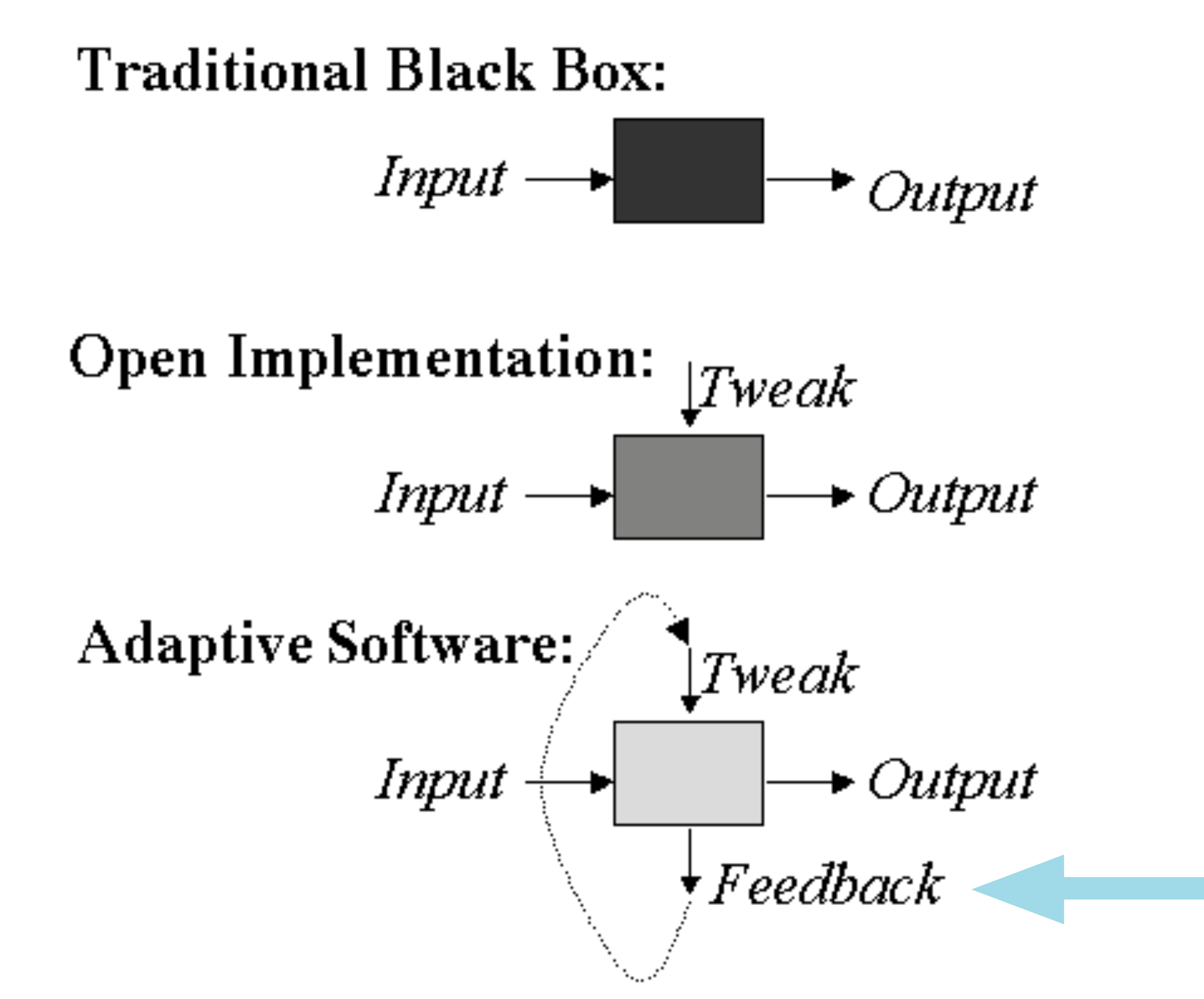
Do not forget

- Functional programming
  - would deserve an entire talk
  - Is a central component of the new data software architecture
    - Divide & conquer large data !
    - Micro services are distributed FP !

Do not forget

- Adaptive Software
  - would deserve an entire talk too

Norvig and Cohn 1998



Learnt from (big)data
Machine Learning