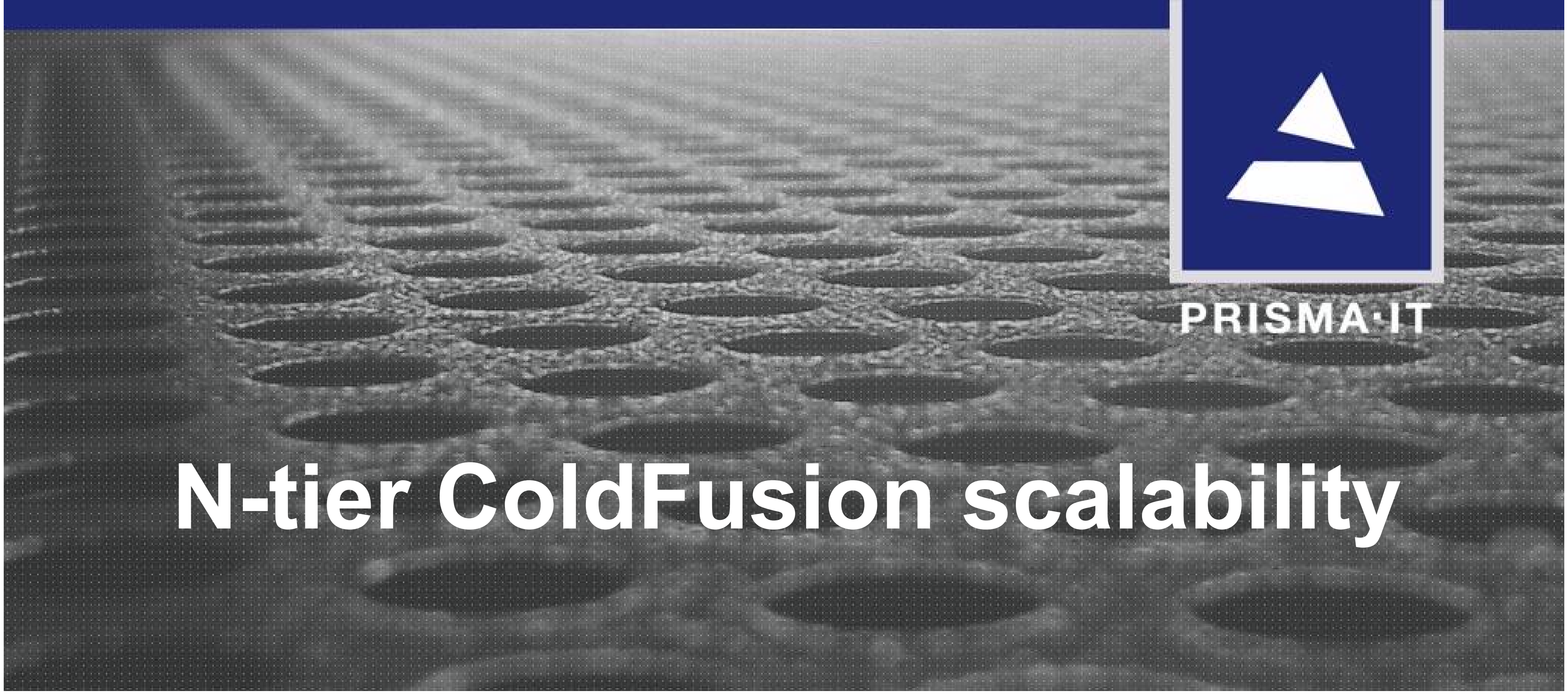


N-tier ColdFusion scalability

About me

- ColdFusion developer for over 10 year
- Adobe Community Expert for ColdFusion
- CTO for Prisma IT in the Netherlands
  - consultancy
  - development
  - hosting
  - training
- Find me at
  - http://jochem.vandieten.net/
  - jochem@prisma-it.com
  - cf-talk@houseoffusion.com

What do I do

- Server stuff
    - CF installation, configuration and troubleshooting
    - Webservers
    - Operating systems
    - Clustering
- Infrastructure
    - Networking
    - Firewalls & loadbalancing
    - DNS
- CF training
- CF development

What do you do?

- developer
- system administrator
- management

Which webserver do you use?

- IIS
- Apache
- other

How many tiers have you used?

- 1
- 2
- 3
- 4
- 5
- 6
- more

Contents

- Definitions
- Tiers
- Scaling the tiers
- HTTP caching

Scalability

A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a scalable system.

http://en.wikipedia.org/wiki/Scalability

Speed and scalability

- are fundamentally different
- speed is how fast a single request is processed
- scalable system do not have to be fast
- fast systems are not necessarily scalable

Scaling dimensions

- scaling up
  - add more resouces to a node
  - CPUs
  - RAM
  - suited for tightly interconnected processes
- scaling out
  - add more nodes to a system
  - add a computer
  - popularized by commodity hardware
  - suited for loosely interconnected processes

Partitioning dimensions

- horizontal partitioning
  - every node does the same as every other node
  - it just does it for another domain
    - East Coast
    - West Coast
- vertical partitioning
  - every node does a specific task
    - webserver
    - application server
    - database

Tiering semantics

- traditional three tier architecture
  - presentation tier
  - business tier
  - data tier
- three tier web architecture
  - web server
  - application server
  - database server

Tiering semantics

- client applications
- proxies
- web server
- application server
- database server

every web application is tiered because the client runs elsewhere

Reasons for N-tier architectures

- speed
- scalability
- security
  - penetrating is like peeling an union
  - every tier has wel defined properties
- manageability
  - every tier has wel defined properties
  - every tier is individually testable

Scaling the client tier

- completely independent users
- users bring their own resources
- perfect scaling out model
- but not under your control

Scaling the proxy tier

- mobile operators and large corporations have their own proxies
- Content Delivery Networks offer proxies for hire
- you can operate your own proxies
- they are completely independent of eachother
- scaling out is trivial

Scaling the webserver tier

- scaling out is quite easy
- the main problem is getting the content on every server
- use a shared / network drive
- use file system replication

Scaling the ColdFusion tier

- scales up good
  - except for createUUID()
  - createUUID() generates a fixed number of UUIDs per second
- scales out OK
  - problem is in memory variables
  - sticky sessions is an accepted solution

Partitioning the ColdFusion tier

- set up specific processes (instances) for specific tasks
- make the task asynchronous
- you have used this already: mail spooling
- it gets better when you do this on a larger scale

Partitioning the ColdFusion tier

- long running processes to partition
  - reporting
  - indexing
  - PDF generation
- set up a webservice to offload this to another instance
- tune your hardware for the task at hand
  - switch off hyperthreading for PDF generation
    - faster response times
    - same throughput
- tune ColdFusion for the task at hand
  - lower the number of simultaneous threads (3 per core)
  - don't save class files

Scaling the database tier

- scales up pretty good
  - the database is usually the biggest box
- scaling out has consequences, pick one
  - no transactional integrity
  - read-only nodes
  - lag
  - n^3 increase of (death)locks

What makes an infrastructure scalable?

Every tier handles fewer requests then the previous tier

Every tier handles fewer requests

- proxy gets all requests
- web server gets non-cached requests
- application server gets non-cached dynamic requests

HTTP caching

- intended to eliminate requests
  - based on content expiration
  - upstream tells downstream: use this without further questions until Y
- intended to eliminate full responses
  - based on content validation
  - downstream asks upstream: I have version X is that OK

Eliminating full HTTP responses

- the browser tells the server which version it has
  If-Modified-Since: Sat, 09 Sep 2006 16:21:52 GMT
- the server tells the browser what it sends
  ETag: Mon, 17 Mar 2008 02:00:00 GMT
  Last-Modified: Mon, 17 Mar 2008 02:00:00 GMT
- the server tells the browser it has not changed
  HTTP/1.1 304 Not Modified
  ETag: Mon, 17 Mar 2008 02:00:00 GMT
  Last-Modified: Mon, 17 Mar 2008 02:00:00 GMT

ColdFusion caching

ColdFusion will not automatically add cache headers

```
HTTP/1.x 200 OK
Date: Tue, 20 May 2008 11:44:02 GMT
Server: Apache/2.2.8 (Win32) JRun/4.0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Apache caching

Apache will automatically add cache headers

```
HTTP/1.x 200 OK
Date: Tue, 20 May 2008 11:46:28 GMT
Server: Apache/2.2.8 (Win32) JRun/4.0
Last-Modified: Sat, 20 Nov 2004 18:16:26 GMT
Etag: "2300000027857-2c-3e9549f1af280"
Accept-Ranges: bytes
Content-Length: 44
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

IIS caching

PRISMA·IT

IIS will automatically add cache headers

```
HTTP/1.x 200 OK
Server: Microsoft-IIS/5.1
X-Powered-By: ASP.NET
Date: Tue, 20 May 2008 11:50:15 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Tue, 18 Mar 2008 09:06:48 GMT
Etag: "045d65d788c81:911"
Content-Length: 30778
```

Making caching work

- split static and dynamic content in different files
  - caching ColdFusion generated content is a manual process that you have to work for
  - caching web server content is partially automatic
- split static and dynamic content in different hosts
  - use a CDN
  - no cookies or parameters
- static content generation is a form of caching too!

What pays of most?

If all tiers are equally resource constrained

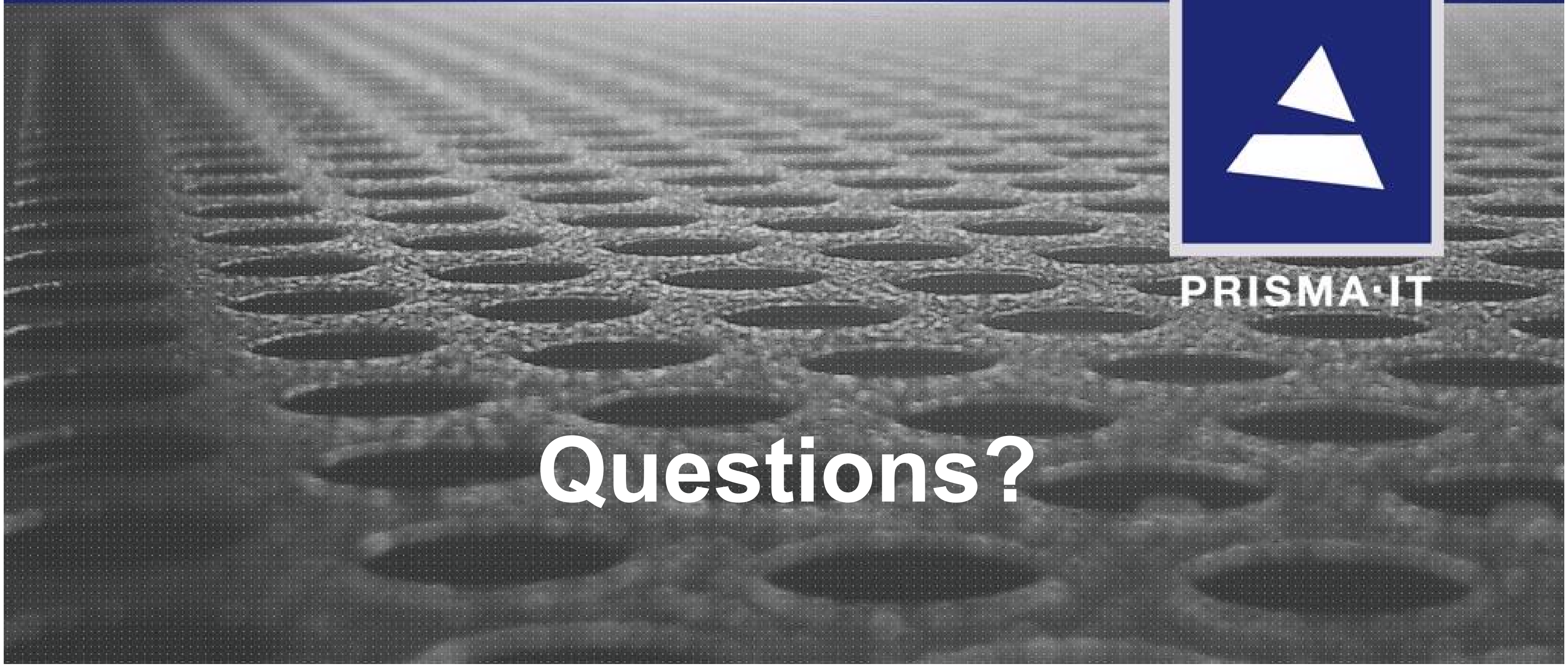
1. moving work from ColdFusion to the webserver
   - if it doesn't reach ColdFusion, it doesn't read the database either
2. moving work from the database to ColdFusion
3. moving work from the webserver to the proxies

Is it worth it?

Imagine the following architecture:

- 2 datacenters
- source based routing
- loadbalancers
- reverse proxies
- webservers
- 2 CF standard servers
- 2 database servers
- 1 backend CF server

How many hits per minute?



Questions?

Security

Ideally tiers communicate with only one protocol for security reasons

- webserver and appserver only HTTP
- appserver and database server only SQL carrier (TDS, NetIQ etc.)
  - this is why you don't want a filesystem database
- if appserver has access to webserver filesystem, the onion is easier to peel
- if appserver has access to database filesystem, the onion is easier to peel

Connecting the webserver and the appserver

- Apache mod_proxy
- JRun connector
- distributed mode
- source on both ends or only on appserver

Apache mod_proxy

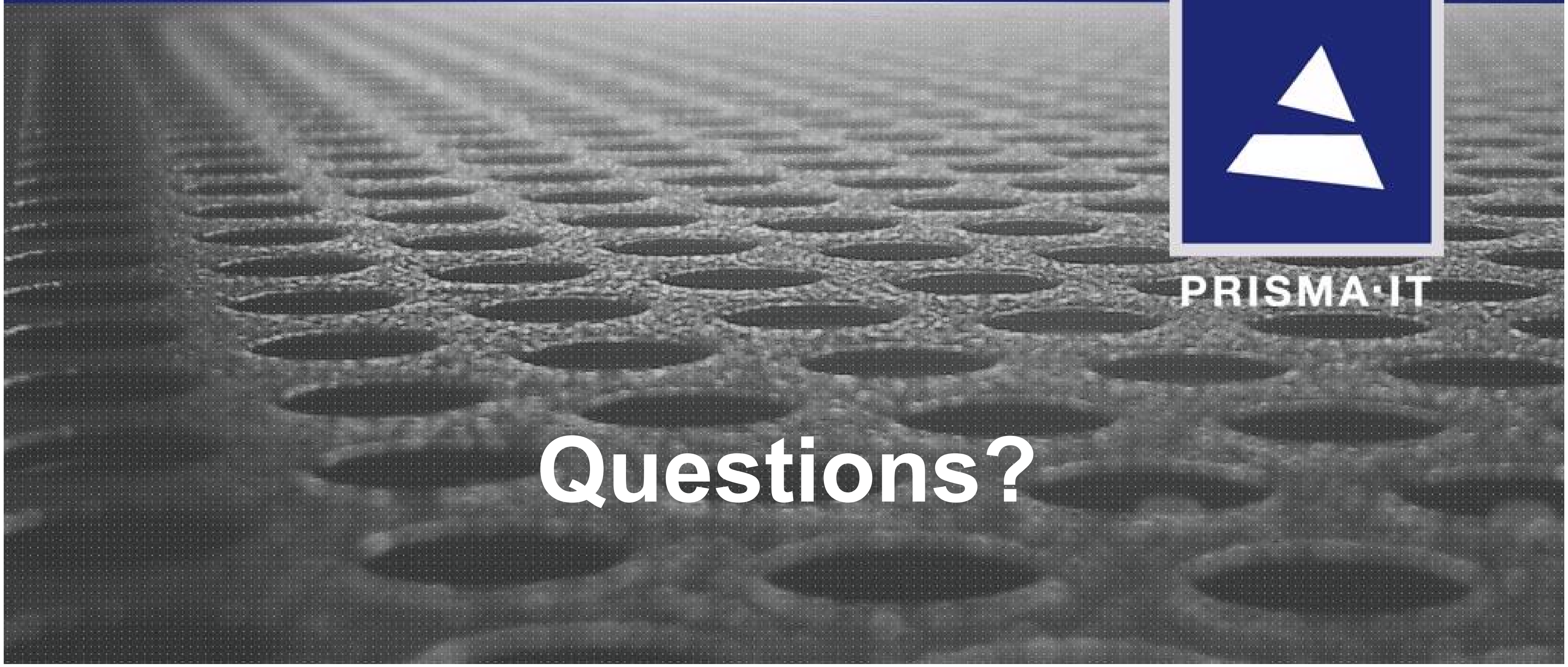
- selectively forward *.cfm
- fully configurable
- no ready-made recipes
- executable code only on appserver

JRun connector

- plug in to webserver
- split configuration
    - webserver
    - connector .ini
    - web.xml on JRun
- ties in with clustering

Distributed mode

- needs partial JRun install on webserver
- ties in with clustering
- officially needs source on both servers
- but you can work around that
    - front controller pattern -> only put index.cfm on the webserver
    - use wildcard connector and web.xml manipulation



Questions?