360技术嘉年华

如何从代码层提高产品质量

360效率平台 张娜

360

目录

目录

产品代码的漏洞检查-Why

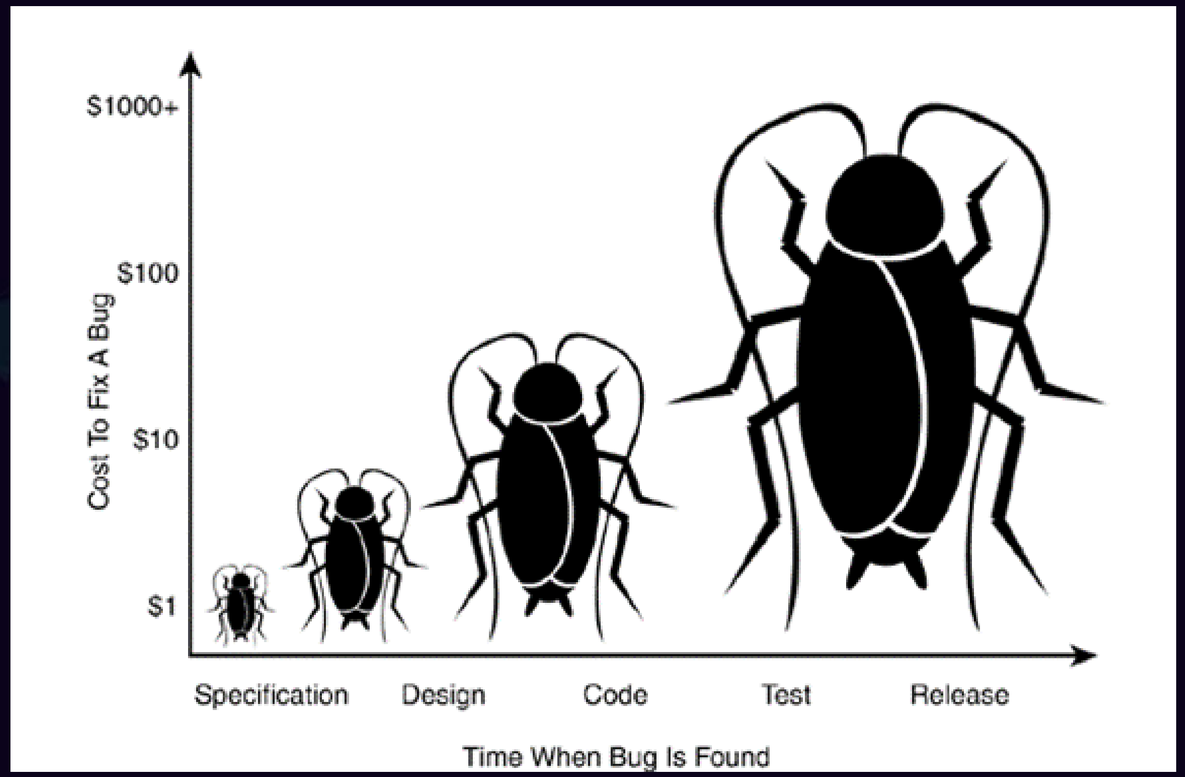
产品质量的问题多数与程序代码相关

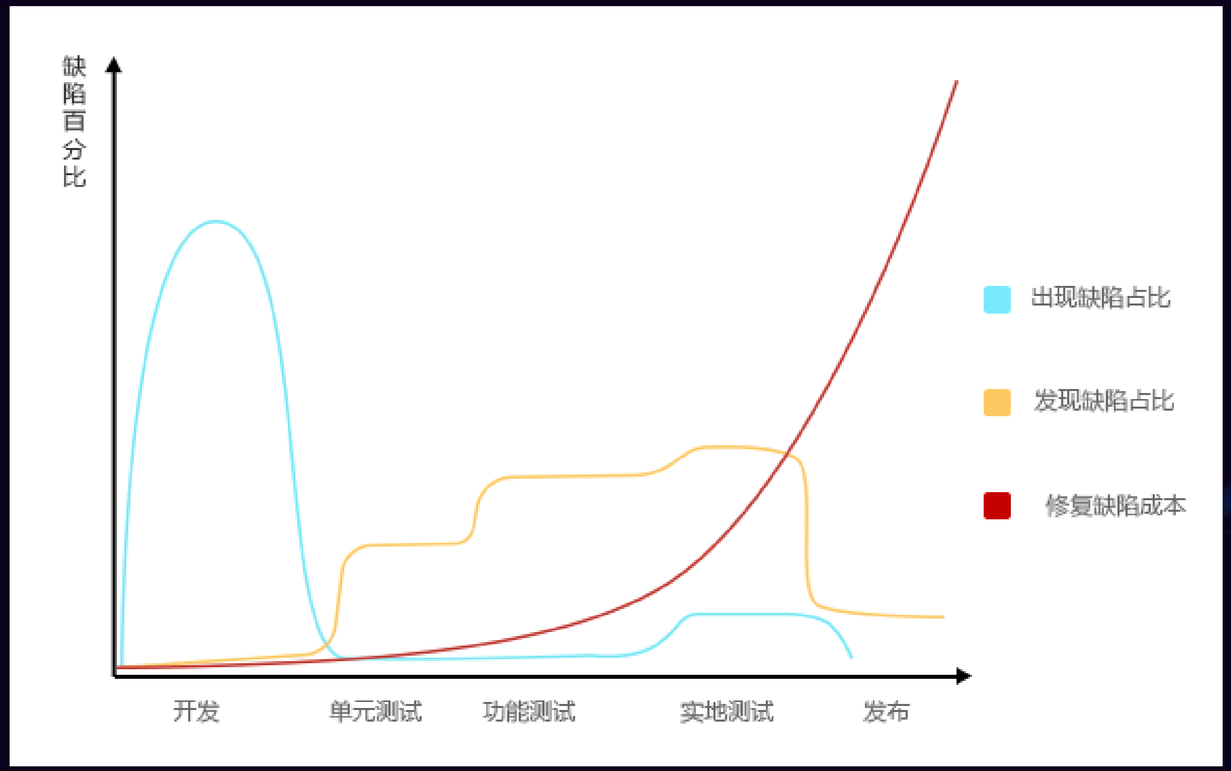
代码的漏洞检查与分析可以帮助用户从根源上减少70%-80%的产品崩溃和安全性问题

只有代码中的崩溃和安全缺陷得以及时消除，最终形成的产品才能具备较高质量，有效降低整体产品风险

产品代码的漏洞检查-When

漏洞发现的越早，修复成本越低





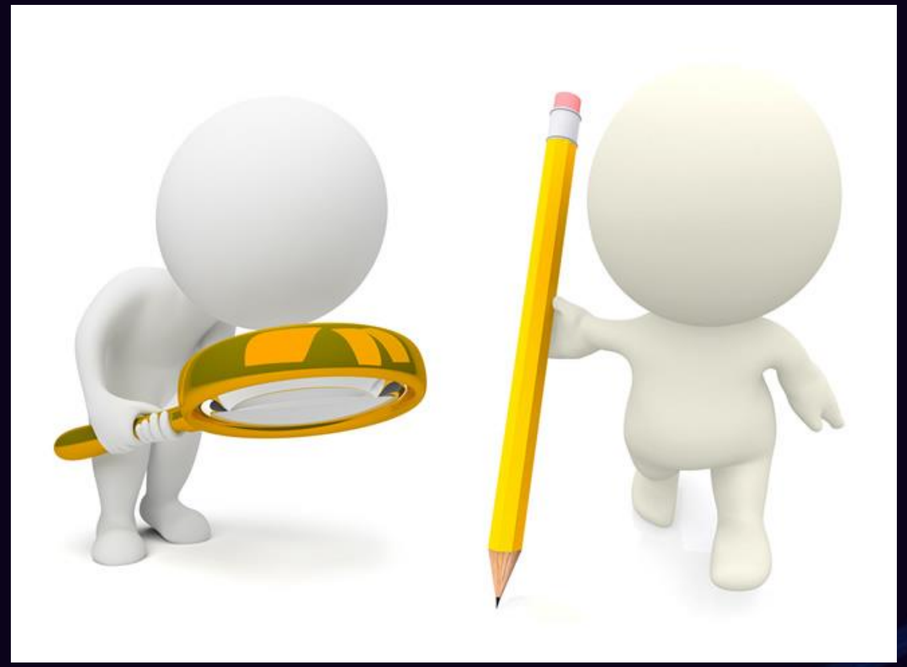
产品代码的漏洞检查-how

✓ 源代码的漏洞扫描与检查

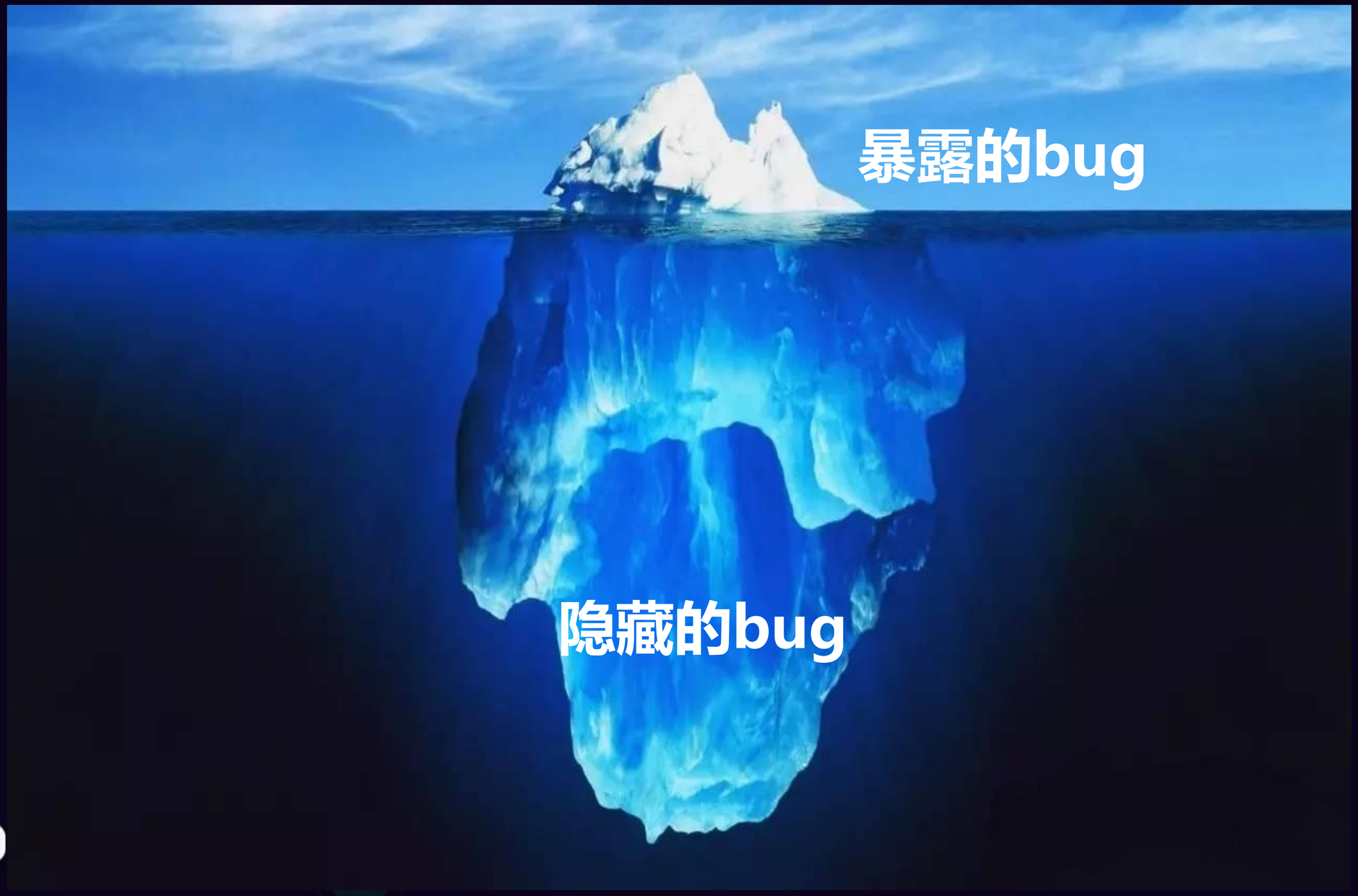
- 自定义代码规范的制定与实时更新
- 具体业务场景的代码规范的制定

✓ 二进制文件的漏洞扫描与检查

- 基于中间代码的漏洞发现
- 二进制代码向中间代码转换的语义恢复

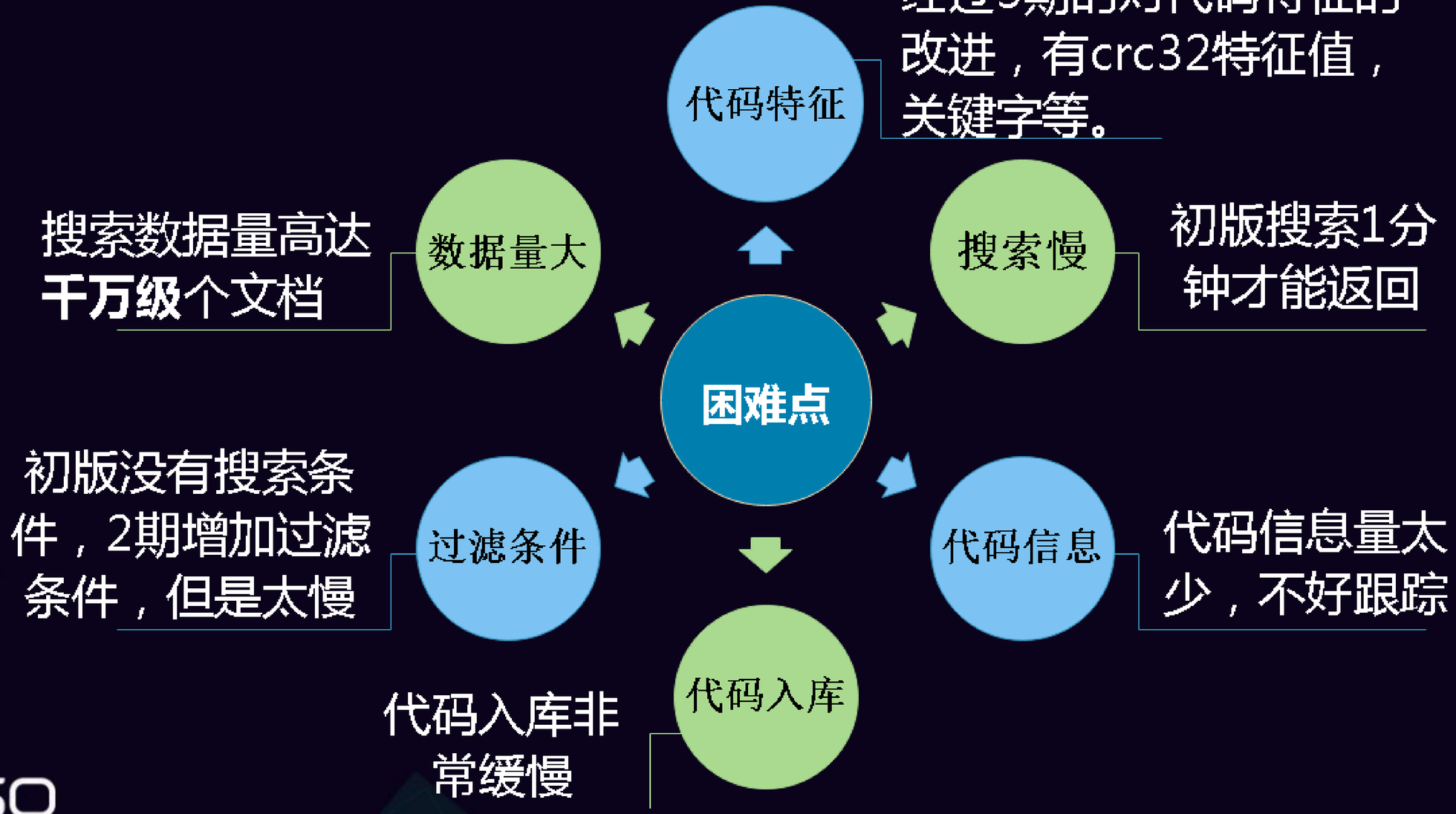


代码漏洞的搜索深挖技术

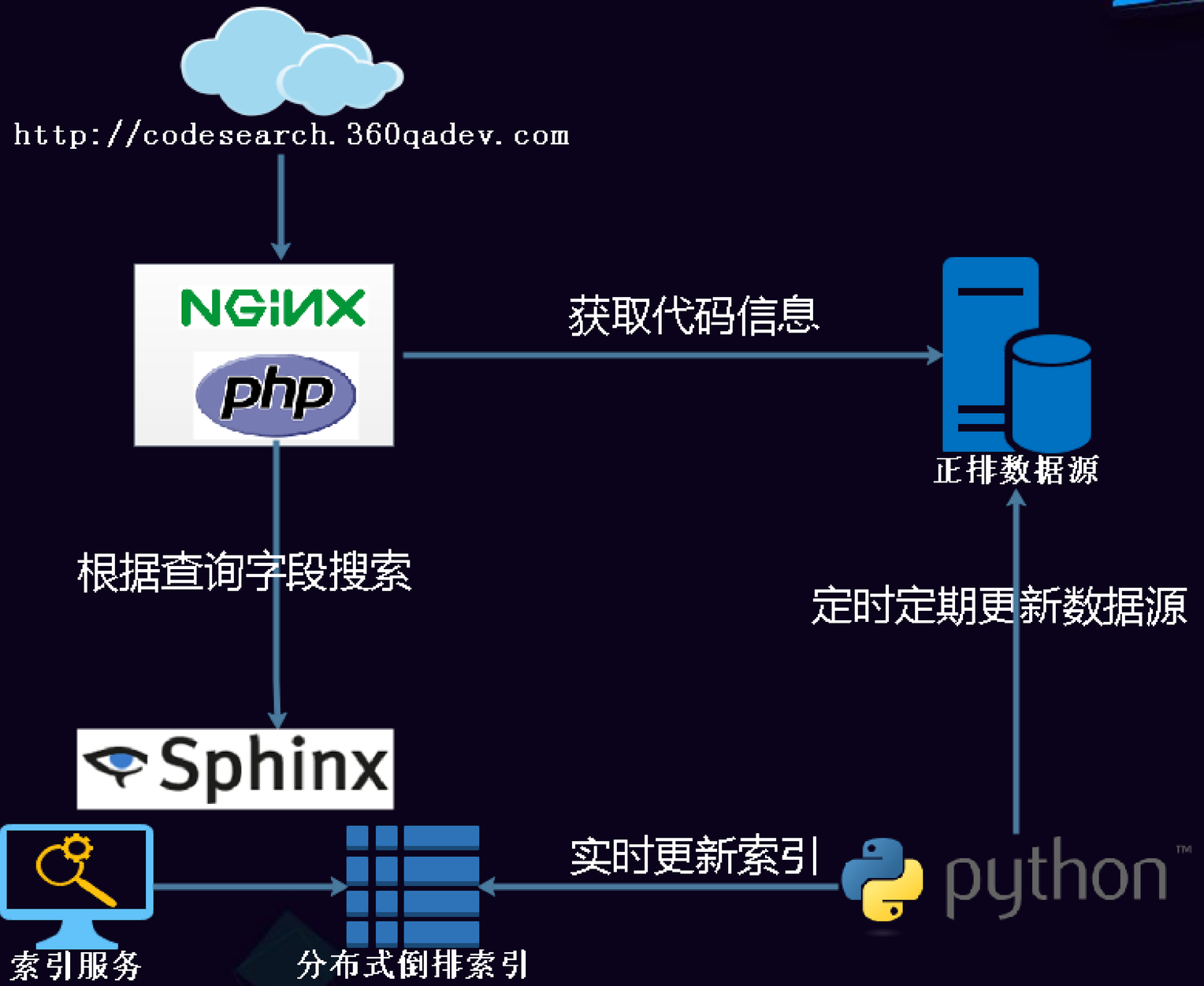


目录

代码搜索的问题与挑战



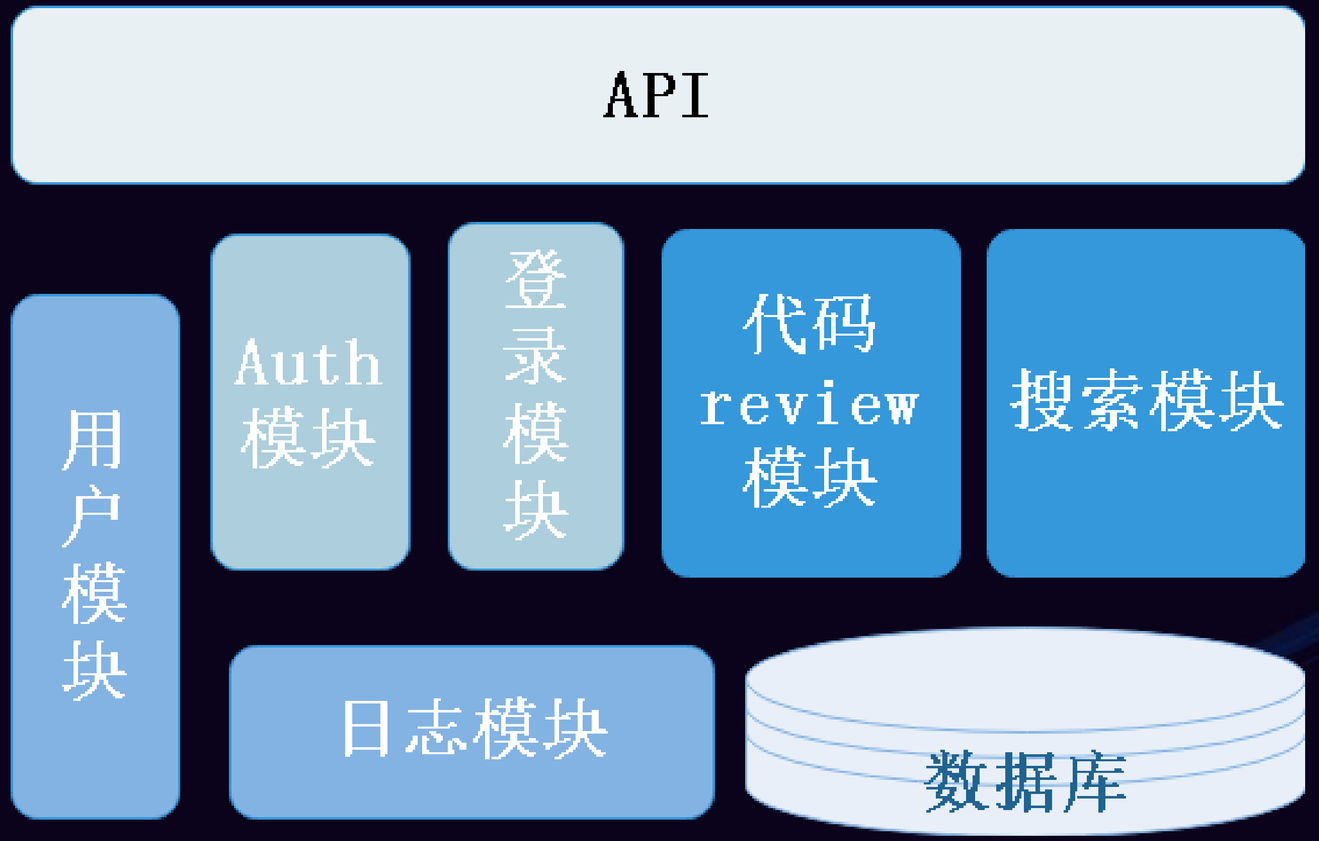
代码搜索的技术架构


http://codesearch.360qadev.com
NGINX
php
获取代码信息
正排数据源
根据查询字段搜索
定时定期更新数据源
Sphinx
实时更新索引
python
索引服务
分布式倒排索引

服务端

服务端为前端或其他系统提供API接口，一共6大模块。

- 搜索模块
- 登录模块
- 校验Auth模块
- 用户模块
- 日志模块
- 代码review模块



数据库为6大模块提供数据支持。

后台

服务

排序

db

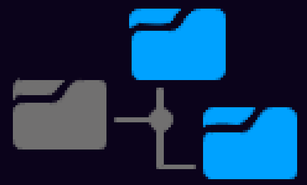
结果数据

索引

mysql

分词模块

倒排索引



数据来源

svn

git

qbuild

授权系统

代码
日志

增量
下载

mysql

数据源增量入库方案

Svn命令：

```
svn log -r {0} --xml -v "{1}" --username "{2}" --password "{3}" --non-interactive --no-auth-cache --trust-server-cert> {4}
svn export -r {0} "{1}" "{2}" --force --username {3} --password "{4}" --non-interactive --no-auth-cache --trust-server-cert
```

Git命令：

```
git log --name-status
```

数据源增量入库方案

Svn（路径包含重复）：
http://svn.example.com/svn/testxxx/111/222/333
http://svn.example.com/svn/testxxx/111
Git（分支重复）：
http://git.example.com/root/11 分支：master
http://git.example.com/root/11 分支：v1.1

去重方法：
模块id+revision
模块id+revision+location（防止程序以外中断）

实时分布式索引技术

- sphinx可扩展性高达数十亿个文档
- 数TB的数据和每秒数千个查询
- 支持各种数据源，xml，sql，python等数据源。
- 支持结果的各种过滤聚合功能
- 快速高效索引
- 应用场合广泛：维基百科，优酷土豆（国内），github

☐ include secondary database models　　18 systems in ranking, July 2019

Rank Jul 2019	Rank Jun 2019	Rank Jul 2018	DBMS	Database Model	Score Jul 2019	Score Jun 2019	Score Jul 2018
1.	1.	1.	Elasticsearch	Search engine, Multi-model	148.81	-0.01	+12.59
2.	2.	2.	Splunk	Search engine	85.49	+0.87	+16.25
3.	3.	3.	Solr	Search engine	59.64	-0.84	-1.88
4.	4.	4.	MarkLogic	Multi-model	13.81	+0.30	+2.50
5.	5.	5.	Sphinx	Search engine	6.69	+0.37	-0.10
6.	6.	6.	Microsoft Azure Search	Search engine	6.42	+0.56	+2.03
7.	7.	7.	Algolia	Search engine	4.76	+0.37	+1.19
8.	8.	8.	Google Search Appliance	Search engine	2.95	-0.11	+0.27
9.	9.	9.	Amazon CloudSearch	Search engine	2.74	+0.20	+0.36
10.	↑ 11.	↑ 11.	CrateDB	Multi-model	0.73	+0.01	+0.30
11.	↓ 10.	↓ 10.	Xapian	Search engine	0.73	-0.01	+0.26
12.	12.	12.	SearchBlox	Search engine	0.27	+0.01	+0.03
13.	↑ 14.	↑ 14.	Exorbyte	Search engine	0.06	+0.01	+0.06
14.	↑ 15.	14.	DBSight	Search engine	0.05	+0.01	+0.05
15.	↑ 16.	↓ 14.	Manticore Search	Search engine	0.05	+0.02	+0.05
16.	↓ 13.	↓ 13.	searchxml	Multi-model	0.05	-0.02	+0.02
17.	17.		FinchDB	Multi-model	0.02	-0.01	
18.	18.	↓ 14.	Indica	Search engine	0.00	±0.00	±0.00

实时分布式索引技术

✓Indexer实时索引工具

索引工具，将数据源的数据进行倒排索引，并存储。

eg: /usr/local/sphinx/bin/indexer -c sphinx.conf code

✓Searchd搜索服务工具

搜索服务工具，通过API调用

eg: /usr/local/sphinx/bin/searchd -c sphinx.conf &

✓Search搜索工具

客户端CLI搜索工具，一般只是测试使用。

eg: /usr/local/sphinx/bin/search -c sphinx.conf mykeyword

实时分布式索引技术

实时索引：

- 代码索引无延时
- 没有额外的定时程序更新和合并索引
- 提高搜索精确性和可靠性

分布式索引：

- 资源利用率提高
- 搜索效率提高
- 搜索并发性提高

```
index coderealtime
{
        type = rt
        path = user/local/sphinx/indexer/files/coderealtime
        rt_field = content
        rt_field = xxxxxx
        rt_attr_uint = xxxxxx
        rt_attr_timestamp = xxxxxx
}

index codedistributed
{
        type = distributed
        local = coderealtime
        agent = localhost:9312:xxxxxx1
        agent = localhost:9312:xxxxxx2
}

searchd
{
        listen                  = 9312
        listen                  = 9306:mysql41
        log                     = /user/local/sphinx/indexer/logs/searchd.log
        query_log               = /user/local/sphinx/indexer/logs/query.log
}
```

代码搜索排序方法

- 词组评分
- 代码提交时间
- BM25算法

$$score(Q,d) = \sum_{i}^{n} W_i \cdot R(q_i,d)$$

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$$R(q_i,d) = \frac{f_i \cdot (k_1 + 1)}{f_i + K} \cdot \frac{qf_i \cdot (k_2 + 1)}{qf_i + k_2}$$

$$K = k_1 \cdot (1 - b + b \cdot \frac{dl}{avg(dl)})$$

$$score(Q,d) = \sum_{i}^{n} IDF(q_i) \cdot \frac{f_i \cdot (k_1 + 1)}{f_i + k_1 \cdot (1 - b + b \cdot \frac{dl}{avg(dl)})}$$

目录

提高产品质量方法

✓ 结合业务督促开发修复代码漏洞

- 督促开发修复代码搜索中出现的漏洞，去除产品隐患
- 结合业务搜索相关代码漏洞，避免代码漏洞的扩散

✓ 产品代码的敏感词检查

- 代码审计系统的敏感词和禁用api的检查
- 文件签名系统的敏感签名信息的检查

目录

总结与展望

总结：

- 能够快速定位问题
- 搜索速度显著提升（联合&过滤）
- 搜索排序质量提高
- 辅助优化产品代码质量

展望：

- 代码推荐结合代码语义上下文和AI的方法
- 函数式的代码推荐

谢谢！