任务调度架构变迁史

知道创宇
杨冀龙

需求1—检测一个URL

需求

- 对1个给定的URL检测是否有XX问题
- 输出信息到屏幕

实现

- Xxcheck.py –u url
- 关键内部函数checkXX(url)完成检测工作

需求2—检测一批URL

需求

- 对给定URL列表文件中URL进行XX问题检测
- 信息记录到日志文件，URL级别为万

实现

- Xxcheck.py –f urls_file

需求3—多线程，有进度信息

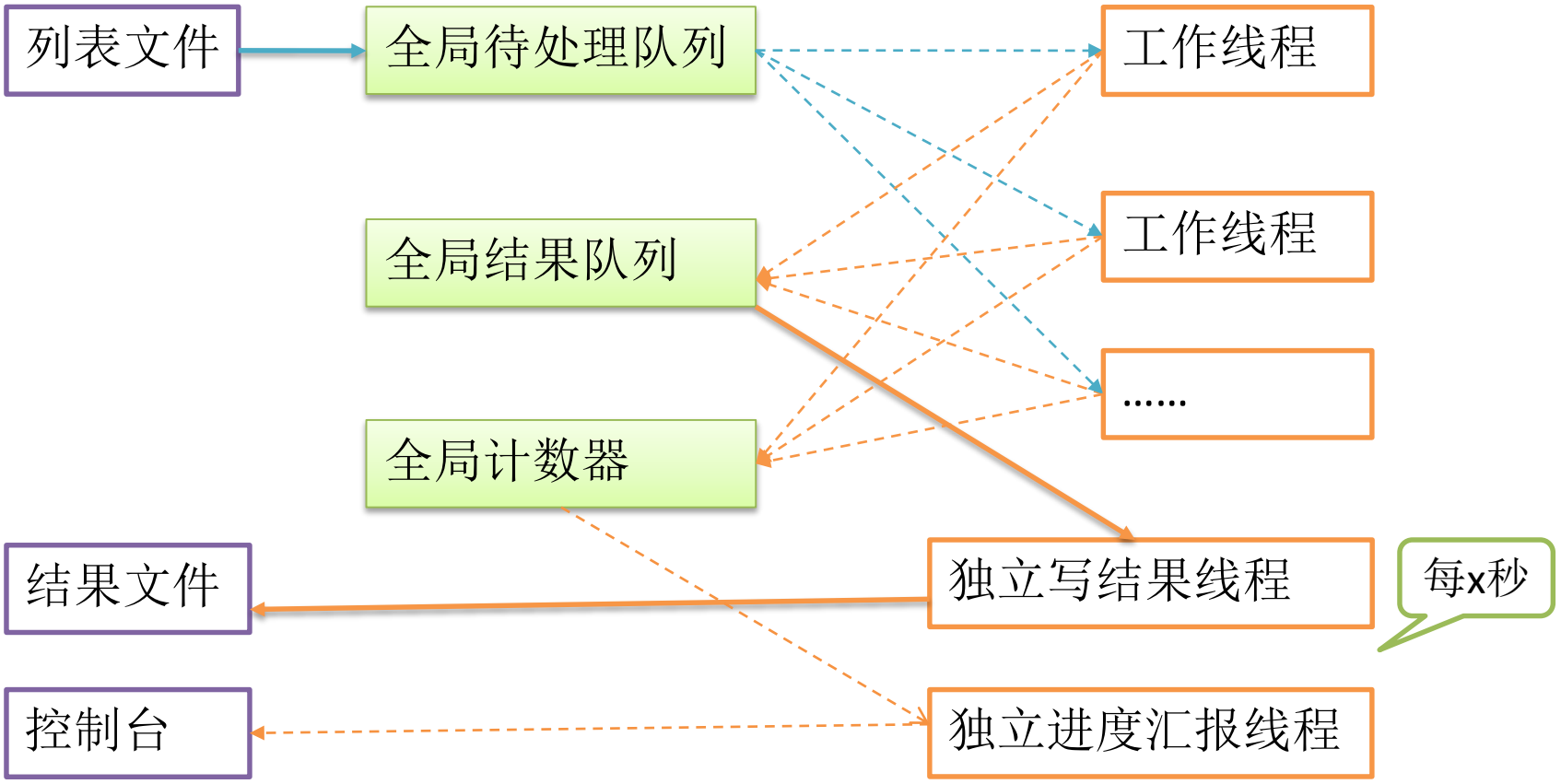
需求

- 为了提高速度，需要使用多线程
- 屏幕上需要有进度信息

实现

- Xxcheck.py –f urls_file –t thread_number
- 多个线程进行同步检查工作，一个独立线程显示进度

需求3架构


列表文件
全局待处理队列
工作线程
工作线程
全局结果队列
.....
全局计数器
结果文件
独立写结果线程
每x秒
控制台
独立进度汇报线程

需求4—多进程，写DB，报进度

需求

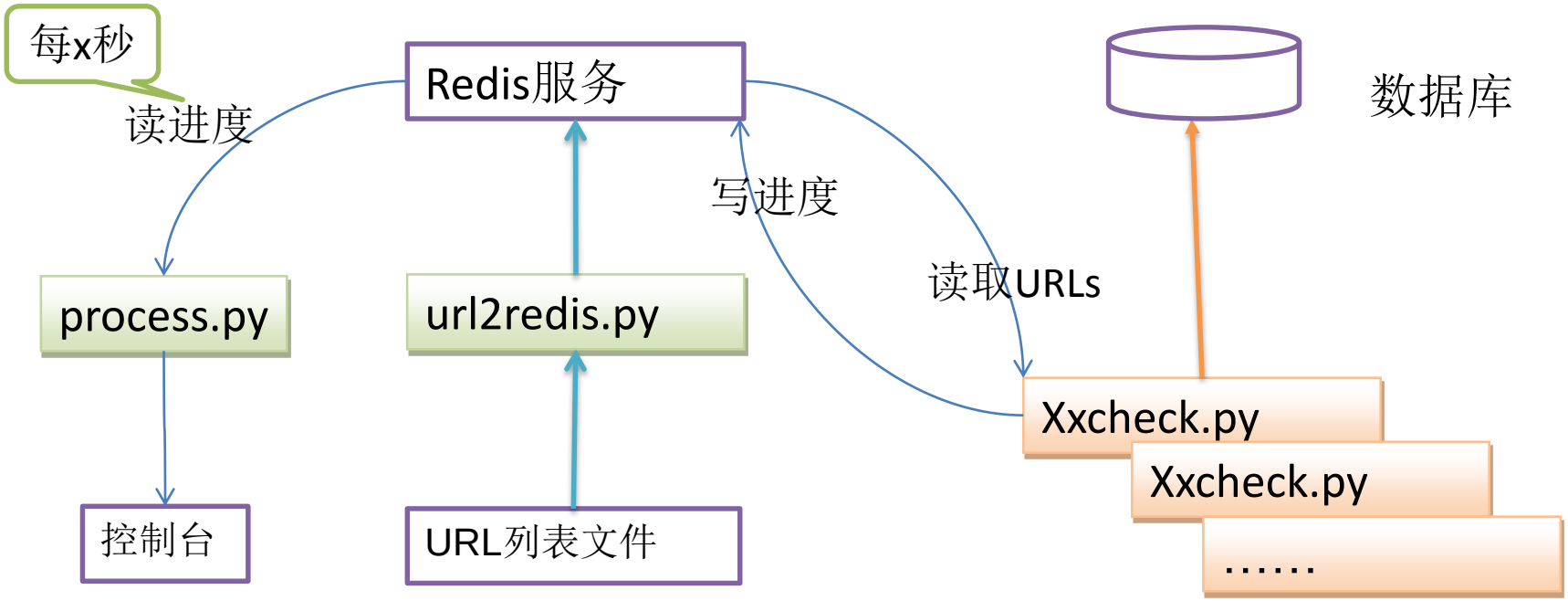
Python线程不能真正全部发挥所有CPU系统资源，线程数上限大约200，需要进一步提高速度，检测结果写入数据库，URL数量为10万，工作中打印进度信息，结束时统计

实现

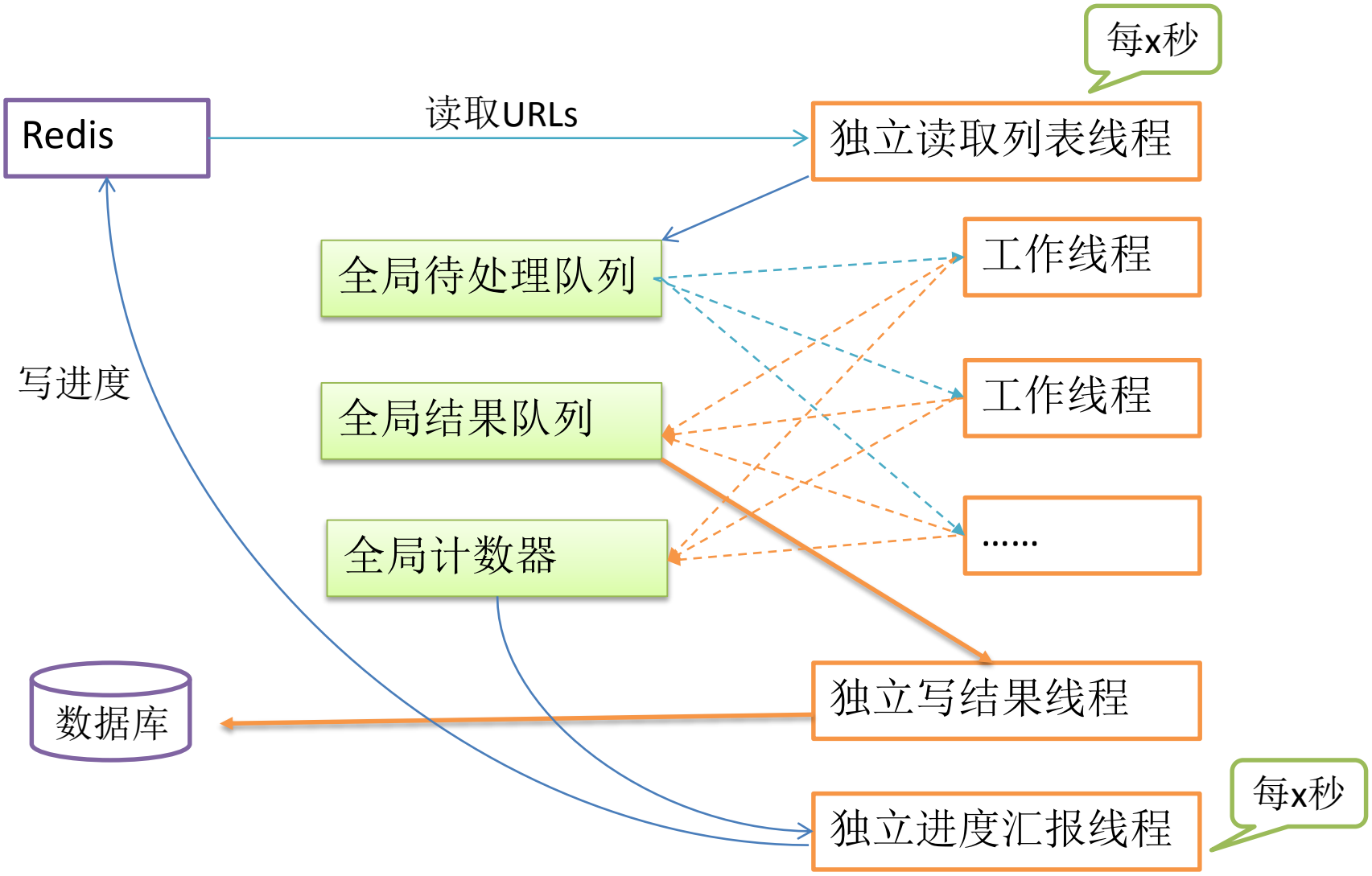
Xxcheck.sh
内部调用url2redis.py
后台启动多个xxcheck.py
启动一个进程，从redis中读取进度并打印

需求4架构

整体架构


每x秒
读进度
Redis服务
数据库
写进度
读取URLs
process.py
url2redis.py
Xxcheck.py
Xxcheck.py
……
控制台
URL列表文件

需求4架构


每x秒
Redis
读取URLs
独立读取列表线程
全局待处理队列
工作线程
写进度
全局结果队列
工作线程
全局计数器
.....
数据库
独立写结果线程
独立进度汇报线程
每x秒

需求5—分布式

需求

单主机资源用尽，需要同时能够在多个服务器上并行进行检测任务；如果一个检测进程崩溃，能够自动侦测并将任务转发给其他进程。

实现

？

The end。