



集算器

# docker下集算器多租户方案

润乾软件出品



本文主要介绍了集算器在docker环境下运行所需要的服务器环境设置、集算器客户端的使用及几个典型的dfx应用案例说明。

docker管理员导入集算器镜像后，需要配置docker管理服务，用户的磁盘、内存、CPU资源配置管理等。

用户在client端登陆成功后，可调用远程dfx脚本进行计算，也可使用集算器集群进行计算业务。



- 1、序言
- 2、结构
- 3、服务器环境设置
  - A、安装docker
  - B、安装集算器镜像
  - C、资源配置管理
    - I、配置文件列表
    - II、配置文件应用
    - III、容器目录映射
    - IV、docker访问资源控制
  - D、docker服务管理
    - I、文件列表
    - II、server配置文件
    - III、docker用户管理
    - IV、启动docker服务

- E、集算器节点配置
- F、集算器配置
- G、从服务器配置
- 4、客户端使用
  - A、用户登录
  - B、集算器资源配置
  - C、文件数据访问
- 5、其它数据源
  - A、数据库的使用
  - B、外部库的使用
- 6、集群的使用
- 7、总结

# 1、序言

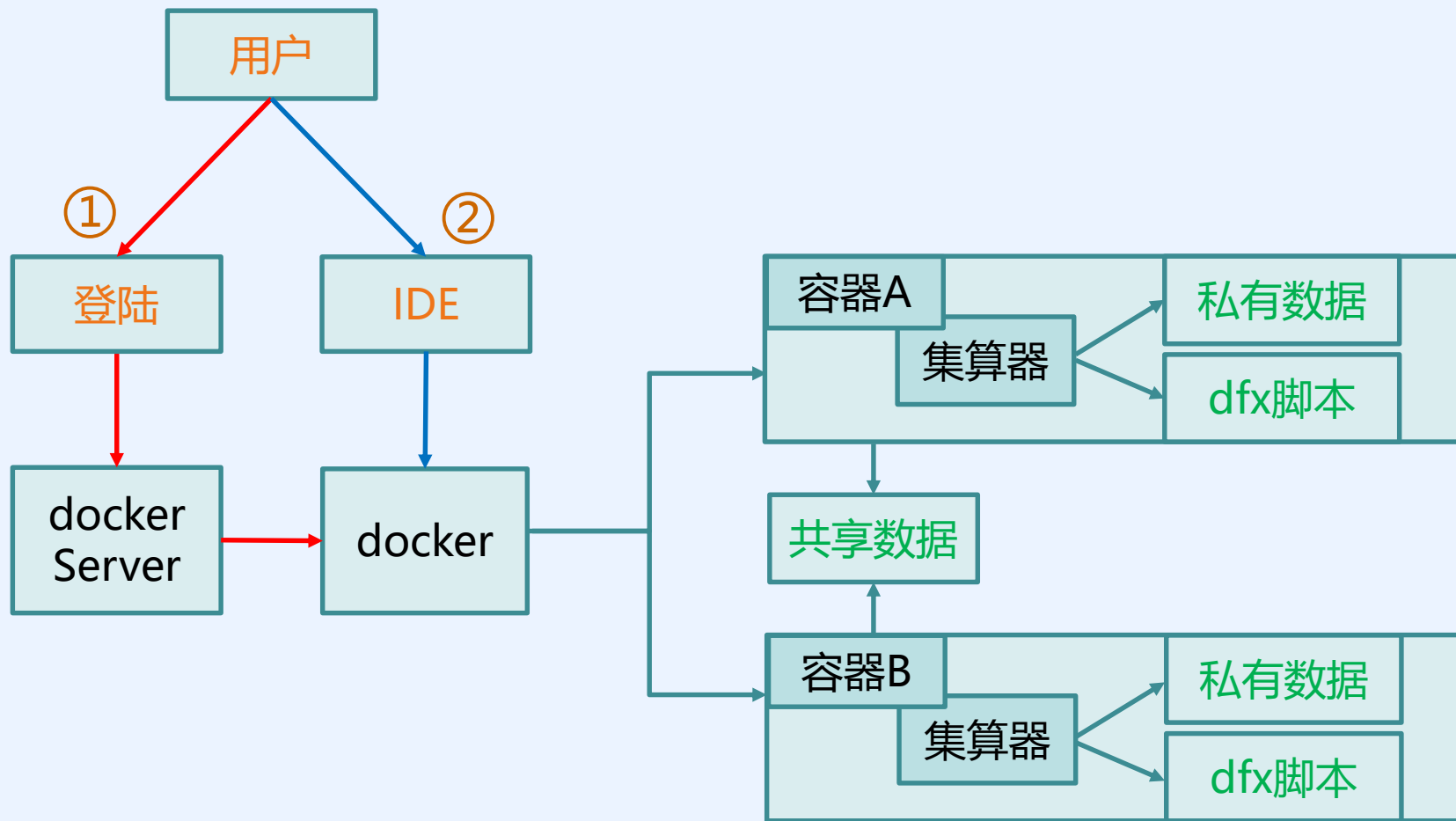


docker是容器虚拟化技术，从文件系统、网络互联到进程隔离等，极大的简化了容器的创建和维护，使得 docker 技术比虚拟机技术更为轻便、快捷。它可以有多个容器且相互隔离，互不影响。

集算器可以部署到服务器上，并组成集群系统，以完成复杂的分析处理工作，增强数据处理能力。

将集算器和docker结合，有利于在同样配置的设备情况下，可同时实现更多不同计算业务需求，且它们之间相互独立，互不干扰，进行高效的运算、合理的资源利用与管理。

## 2、结构



## 2、结构



### 用户工作流程:

- A、**登录**: 客户端用户登陆时, dockerServer根据用户配置启动docker容器、集算器服务。
- B、**工作**: 登陆成功后, 用户通过客户端的集算器IDE访问服务端集算器, 执行服务器上的dfx脚本。
- C、**退出**: 工作完成后, 用户退出时关闭与自己相关的docker容器。

### 服务器、docker容器、集算器关系:

- A、docker容器与集算器是一一对应的关系。
- B、一台服务器上运行多个docker容器
- C、一个用户可以使用一个或多个docker容器。
- D、本应用系统支持多个用户同时操作且互不影响。

**共享数据**: 任何docker用户都可访问的存储目录或文件, 只能读不能改。只能由系统管理员维护。

**私有数据**: 只有登陆的用户才能访问自己的存储目录或文件。每个用户有自己独立的、私有的数据存储区, 其它的用户不能访问。

# 3、服务器环境设置



本应用提供的docker服务只在Linux环境使用, 采用的是C/S结构。现以ubuntu15 + jdk1.8 + docker 1.9.1, 通过root帐号在/home/docker/java下安装说明。

## A、安装docker:

```
# apt-get install docker-engine
```

**查看docker信息:**

```
# docker info
```

**启动docker服务:**

```
# /etc/init.d/docker start
```

**运行测试:**

```
# docker run busybox /bin/echo "Hello World"
```

```
root@master:~# docker run busybox /bin/echo "Hello world"
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox

234382100b69: Pull complete
2ed2a0eb52cd: Pull complete
Digest: sha256:26820e4a4b7b507d71fc0d1983ca00a1a2774e03e590e68875669f0e2a768497
Status: Downloaded newer image for busybox:latest
Hello world
root@master:~# █
```

若没有**安装java**,则安装可用

```
# apt-get install oracle-java8-installer
```

## B、安装集算器镜像



### 下载集算器镜像与docker管理应用文件

```
# wget http://download.raqsoft.com.cn/esproc/docker/esproc.tar.gz  
# wget http://download.raqsoft.com.cn/esproc/dockerServer/dockerServer.zip
```

### 解压镜像压缩文件: esproc.tar.gz

```
# tar -zxvf esproc.tar.gz
```

### 导入esproc镜像:

```
# docker load < esproc.tar
```

### 查看安装镜像:

```
# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
esproc	latest	ea30c50c8893	7 hours ago	783.4 MB

### 测试镜像 (raq\_start.sh来源参考后面的docker服务管理):

```
# cd /home/docker/java  
# ./raq_start.sh aaa 500m 512 /opt/app/aaa 8282 /home/docker/share
```



## B、安装集算器镜像



### 查看服务进程:

```
# ps -ef
```

```
[kworker/3:1]
docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8281 -container-ip 172.17.0.2 -container-port 8281
/bin/bash
/bin/bash /raqsoft/esProc/bin/ServerConsole.sh -pllocalhost:8281
java -Xms128m -Xmx1024m -cp /raqsoft/esProc/classes:/raqsoft/esProc/lib/*:/raqsoft/common/jdbc/* -Duser.language=en
docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8282 -container-ip 172.17.0.3 -container-port 8282
[kworker/1:2]
/bin/bash
/bin/bash /raqsoft/esProc/bin/ServerConsole.sh -pllocalhost:8282
java -Xms128m -Xmx1024m -cp /raqsoft/esProc/classes:/raqsoft/esProc/lib/*:/raqsoft/common/jdbc/* -Duser.language=en
```

**成对进程:** 确定进程docker-proxy、ServerConsole.sh都存在，每个集算器应用在docker容器中都有这两个进程，它们使用同一个port。

若缺少进程ServerConsole.sh，则表示集算器服务启动不成功，可进入docker容器后，通过启动集算器服务看到相关日志信息。

### 查看当前运行的容器:

```
# docker ps
```

## B、安装集算器镜像



### 查看日志:

```
# ./go.sh aaa //用户aaa进入docker容器
# ps -ef //docker中查看进程
# ./ServerConsole.sh -plocalhost:8285 //集算器服务启动
```

```
root@master:/home/docker/java# ./go.sh aaa
root@cdc49cbd17a2:/raqsoft/esProc/bin# ./ServerConsole.sh -plocalhost:8285
Log level:DEBUG
[2019-11-27 09:17:15]
DEBUG: Dfx path: /app/demo

[2019-11-27 09:17:15]
INFO: load library [mongo] from MongoCli

[2019-11-27 09:17:15]
INFO: Inner version: 20191014

[2019-11-27 09:17:15]
INFO: Starting unit server...

[2019-11-27 09:17:15]
INFO: Using start.home=/raqsoft/esProc

[2019-11-27 09:17:16]
DEBUG: Using TempTimeout=12 hour(s).

[2019-11-27 09:17:16]
DEBUG: Using ProxyTimeout=12 hour(s).

java.lang.Exception: Starting 172.17.0.2:8285 failed.Please check does the port is occupied or no license permit.
    at com.raqsoft.parallel.UnitContext.<init>(UnitContext.java:265)
    at com.raqsoft.server.unit.UnitServer.checkMainProcess(UnitServer.java:328)
    at com.raqsoft.server.unit.UnitServer.run(UnitServer.java:421)
    at java.lang.Thread.run(Thread.java:745)

Exit
root@cdc49cbd17a2:/raqsoft/esProc/bin# more ../
```

通过日志可以  
跟踪问题原因

# C、资源配置管理



## I、配置文件列表：

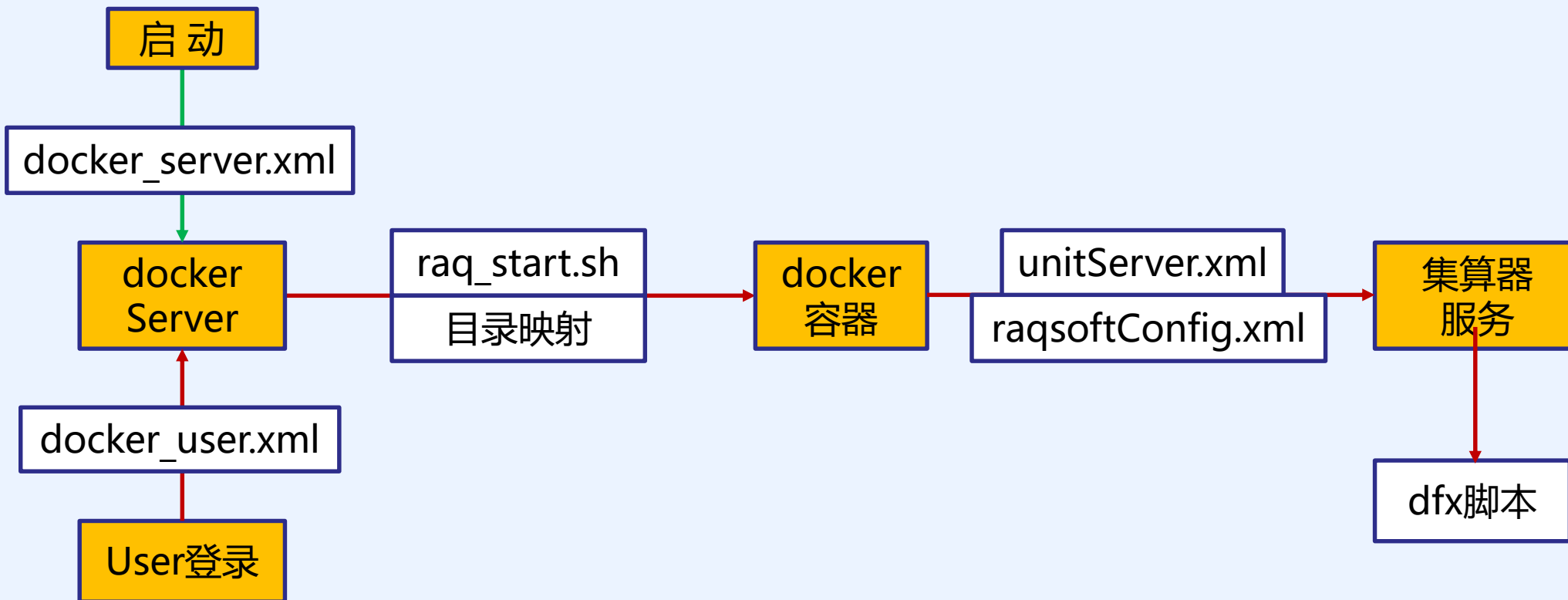
选项	目录	说明
dockerServer	/home/docker/java	docker管理应用程序
docker_server.xml	/home/docker/java	docker服务配置文件，主服务器才有<NodeType>选项
docker_user.xml	/home/docker/java	docker用户管理文件，主服务器才有此文件
unitServer.xml	/home/docker/share	集算器集群资源配置文件
esproc_lic.xml	/home/docker/share	集算器授权文件
user	/opt/app/user	分配给docker用户的目录
raqsoftConfig.xml	/opt/app/user/config	集算器配置文件

/opt/app是用户数据存储所在的目录，用户目录将被映射到docker容器中。

## II、配置文件应用



配置文件在流程环节中的使用。



# III、容器目录映射

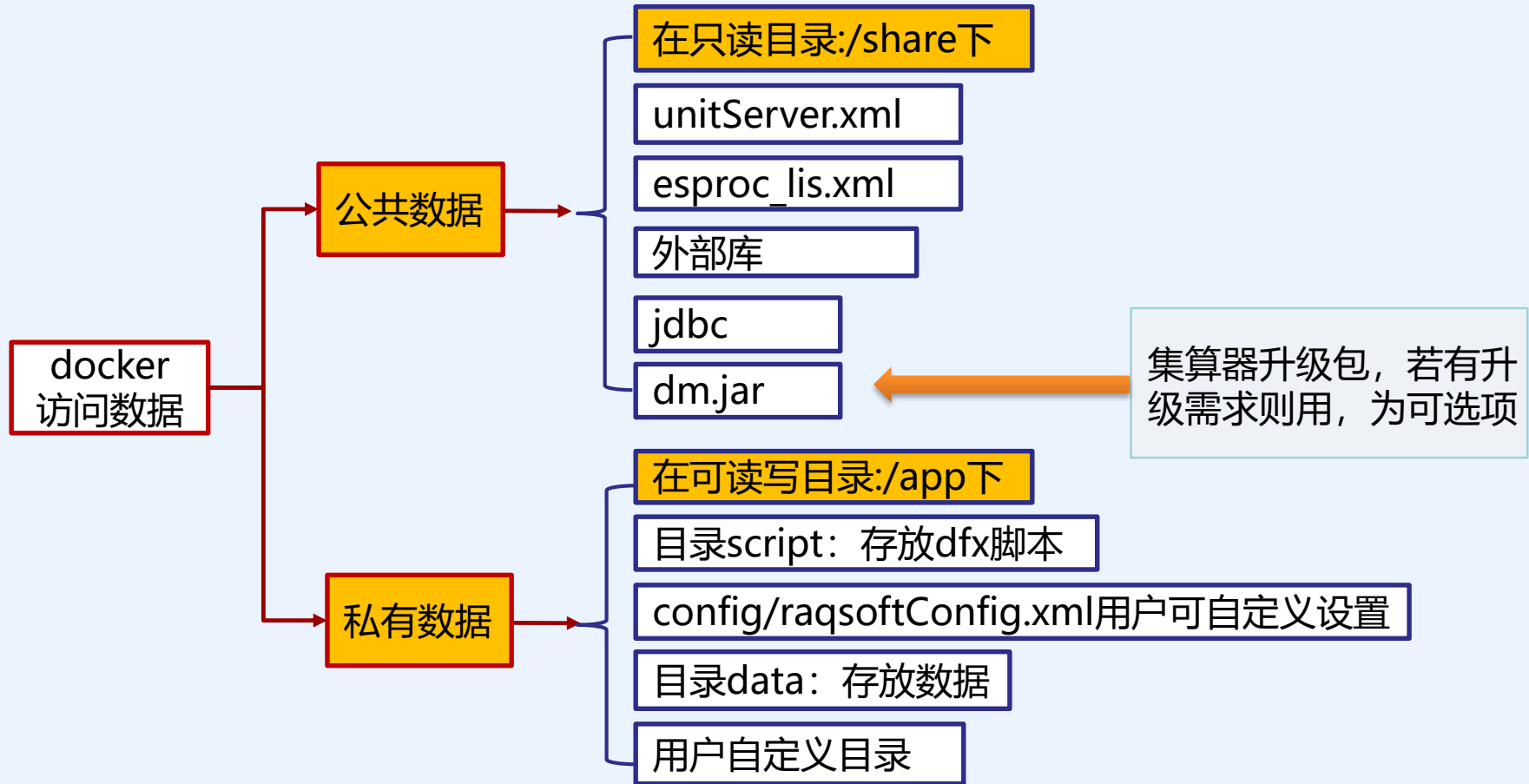


docker镜像制作完成之后，里面的内容是只读不能修改，容器中操作的数据也会随容器关闭而丢失。为了满足用户的需求，减少用户重复操作，实现集算器可读写用户数据，将在宿主机下的共享目录及用户目录挂载到docker容器中。

宿主机目录	Dokcer容器目录	说明
/home/docker/share	/share	全局共享目录
/home/docker/share	/share/jdbc	Jdbc jar文件存放目录
/home/docker/share	/share/extlib	外部库文件存放目录
/opt/app/user	/app	用户主目录<user>
/opt/app/user/script	/app/script	dfx脚本存放目录
/opt/app/user/data	/app/data	数据存放目录
/opt/app/user/config	/app/config	配置文件存放目录

缺省情况下为用户创建了以上目录，当然，用户也可以在<user>下创建子目录，通过“/app/子目录”访问。

# IV、docker访问资源控制



**公共数据:** 由docker管理员管理, 用户为只读访问权限, 所有的用户都能访问。

**私有数据:** docker用户自己管理的数据, 只有本用户才能访问此数据, 用户可读写、修改等, 其它用户不能访问。此数据可持久保存在宿主机目录下, 不会因为docker容器的关闭而丢失。

# D、docker服务管理



dockerServer负责用户登录管理、docker容器启动、关闭管理等。

## I、文件列表：

文件	说明
dockerServer.jar	dockerServer服务应用jar包
go.sh	进入docker容器脚本
raq_get_image.sh	查看正使用的docker容器脚本
raq_start.sh	启动docker 脚本
raq_stop.sh	关闭docker 脚本
server.sh	dockerServer服务启动脚本
docker_server.xml	dockerServer服务配置文件
docker_users.xml	管理docker用户文件

将上述脚本及配置文件来自dockerServer.zip， 放在/home/docker/java目录下。

# D、docker服务管理



## II、server配置文件：

配置文件docker\_server.xml内容如下：

```
<Config>  
  <Server>192.168.0.76:9001 </Server>  
  <Share>/home/docker/share</Share>  
  <NodeType>manager</NodeType>  
</Config>
```

**<NodeType>选项：**设置为主服务器的选项。

**<Server>选项：**服务器ip与port, 也是客户端登录所用的ip与port。

**<Share>选项：** docker容器访问的公共资源目录，比如集算器授权文件，集算器集群配置unitServer.xml文件等，对docker用户来说，本目录是只读属性。



# D、docker服务管理



## III、docker用户管理:

docker用户管理文件`docker_user.xml`, 内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<Users>
  <User>
    <Name>aaa</Name>
    <Alias>张三</Alias>
    <Pwd>123456</Pwd>
    <Memory>1g, 500m</Memory>
    <CPU>512</CPU>
    <Home>/opt/app/aaa</Home>
    <Unit>192.168.0.76:8281,192.168.0.76:8282,192.168.0.77:8281</Unit>
  </User>
  <User>
    <Name>bbb</Name>
    <Alias>李四</Alias>
    <Pwd>123456</Pwd>
    <Memory>100m</Memory>
    <CPU>300</CPU>
    <Home>/opt/app/bbb</Home>
    <Unit>192.168.0.77:8282</Unit>
  </User>
  .....
</Users>
```

# D、docker服务管理



docker管理员为用户分配资源配置信息文件。

**主要内容：**包括用户名、密码及分配给用户的cpu, memory限额，用户目录及集算器所在的节点机。

**CPU 资源限制：**使用--cpu-shares选项进行设置的，实例化镜像时能够对容器进行资源配额设置。

**内存限制：**使用-m参数进行限制，要求只能使用整数，单位为k、m、g。它除了用于设置docker内存参数外，还用于集算器启动时设置java内存(注意：若此参数带小数，则docker或集算器启动会失败)。内存限制参数设置 <Memory> m1, m2 </Memory>, 其中m1为docker内存大小、java启动参数Xms, Xmx选项值，m2为XX:NewSize选项值，为可选项。若m2不写，则m2自动设置为m1值的一半。

**<Home> 选项：**设置用户的目录，docker用户可将数据存放在自己的下。同时docker会将此目录映射到容器中，使得对应的容器可以共享到主机的内容。

**<Unit> 选项：**给用户分配的节点机，**要求ip:port是唯一的**，防止资源上的冲突。每个ip:port对应一个docker容器，可分配给用户一个或多个docker容器。

# D、docker服务管理



## IV、启动docker服务:

**server.sh中修改参数:**

```
start_home=/home/docker/java
```

**启动dockerServer服务:**

```
# /home/docker/java/server.sh
```

**查看dockerServer服务的网络状态:**

```
# netstat -na|grep 9001
```

```
tcp6      0      0 192.168.0.76:9001    :::*           LISTEN
```

# E、集算器节点配置



## 集群资源配置文件unitServer.xml:

```
<SERVER Version="3">
  <TempTimeOut>12</TempTimeOut>
  <Interval>1800</Interval>
  <ProxyTimeOut>12</ProxyTimeOut>
  <Hosts>
    <Host ip="localhost" maxTaskNum="4" preferredTaskNum="2">
      <Partitions>
        <Partition name="0" path="/app/data">
          </Partition>
        </Partitions>
        <Units>
          <Unit port="8281">
            </Unit>
          </Units>
        </Host>
        .....
    <Host ip="localhost" maxTaskNum="4" preferredTaskNum="2">
      <Partitions>
        <Partition name="0" path="/app/data">
          </Partition>
        </Partitions>
```

```
      <Units>
        <Unit port="8290">
          </Unit>
        </Units>
      </Host>
    </Hosts>
    <EnabledClients check="false">
      </EnabledClients>
    </SERVER>
```

docker管理员集中统一分配集算器节点访问端口，让docker访问端口映射到集算器的端口，这样用户通过docker就可以访问集算器了。配置中的Host ip用localhost，每个Host分配一个可用不重复端口。

## F、集算器配置



**集算器的配置文件**raqsoftConfig.xml, 其主要内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config Version="2">
  <Runtime>
    <DBList encryptLevel="0">
    </DBList>
    <Esproc>
      <license>/share/esproc_lic.xml</license>
      <charSet>GBK</charSet>
      <dfxPathList>
        <dfxPath>/app/script</dfxPath>
      </dfxPathList>
      <dateFormat>yyyy-MM-dd</dateFormat>
      <timeFormat>HH:mm:ss</timeFormat>
      <dateTimeFormat>yyyy-MM-dd HH:mm:ss</dateTimeFormat>
      <mainPath>/app/script</mainPath>
      <tempPath></tempPath>
      <bufSize>65536</bufSize>
```

```
<localhost></localhost>
<localPort>0</localPort>
<parallelNum>64</parallelNum>
<zoneLockTryTime>3600</zoneLockTryTime>
<simpleTableBlockSize>1048576</simpleTable
BlockSize>
  <>nullStrings>nan,null,n/a</nullStrings>
  <extLibsPath>/share/extlib</extLibsPath>
    <importLibs>
      <lib>HdfsCli</lib>
    </importLibs>
  </Esproc>
  <Logger>
    <Level>DEBUG</Level>
  </Logger>
</Runtime>
</Config>
```

# F、集算器配置



## 集算器的配置文件说明：

**授权文件：**由于镜像文件是只读的，制作镜像文件时就设置好了缺省的配置。若使用的有效license文件名不是“esproc\_lic.xml”。为了方便操作，管理员将它改成这个文件名并放在share目录下，这样docker启动时就能读取到它。

**dfx 目录：**dfxPath, mainPath设置成/app/script方便调用dfx脚本。

**外部库：**extLibsPath外部库目录设置成/share/extlib，需要加载的外部库用户通过importLibs自定义。

**多线程：**要使用CPU多核优势，parallelNum参数设置为大于1，缺省时它为64。

**友情提示：**配置中的授权文件与path目录，docker容器需要访问它们，一般用户不要修改。要想raqsoftConfig.xml配置生效，需要将它放在用户自己目录config下，然后重启docker

# G、从服务器配置



上面为**主服务器**配置，对于其它**从服务器**配置与它类似，主要包括<资源文件列表>中所涉及到的文件，可将主服务器的配置复制过来，再作适应修改即可。

如将**docker\_server.xml**内容修改如下：

```
<Config>
  <Server>192.168.0.77:9001</Server>
  <Share>/home/docker/share</Share>
</Config>
```

尽管从服务器不需要**docker\_user.xml**文件，但对docker用户而言，从服务器机上的用户目录结构布署与主服务器完全是一致的，他们都是使用同一个**docker\_user.xml**配置文件。

在使用docker前，需要将各个服务器上的**dockerServer**服务启动。

```
# /home/docker/java/server.sh
```

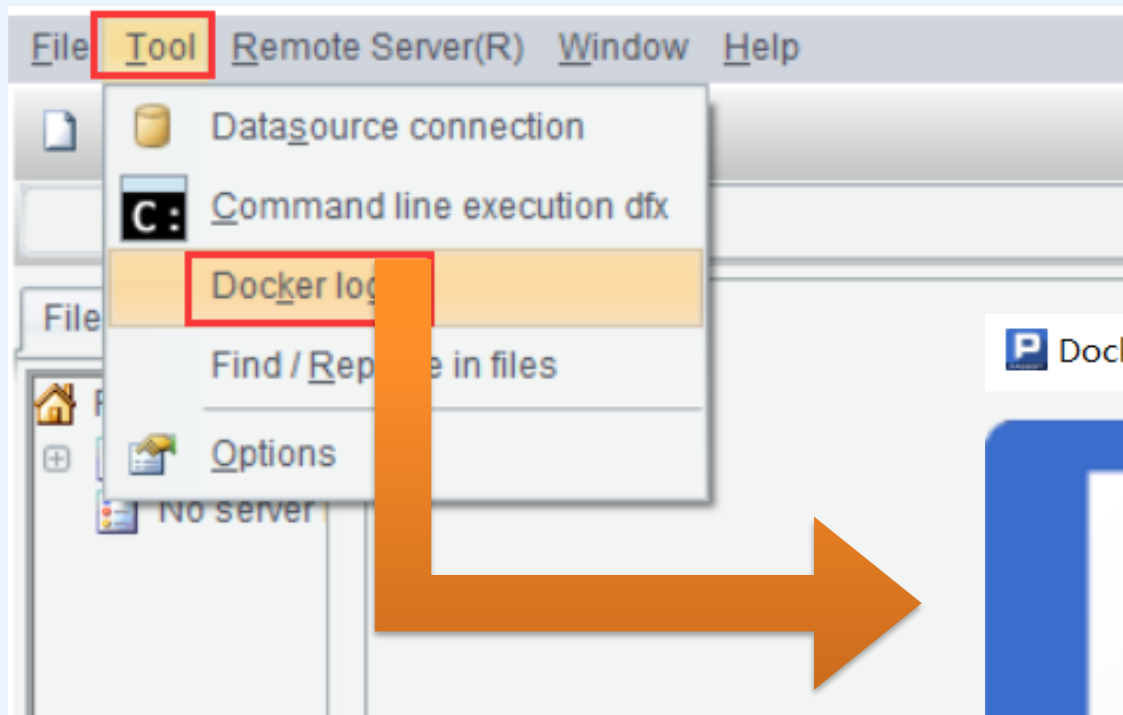
其它安装设置参考主服务器布署。

# 4、客户端使用



## A、用户登录

在集算器IDE中的**菜单->工具->docker登录**



退出也在这里，登录后变成可退出状态

**P** Docker login

IP	192.168.0.76	:Login
Port	9001	Cancel
User name	aaa	
Password	.....	
<input checked="" type="checkbox"/> Remember password		

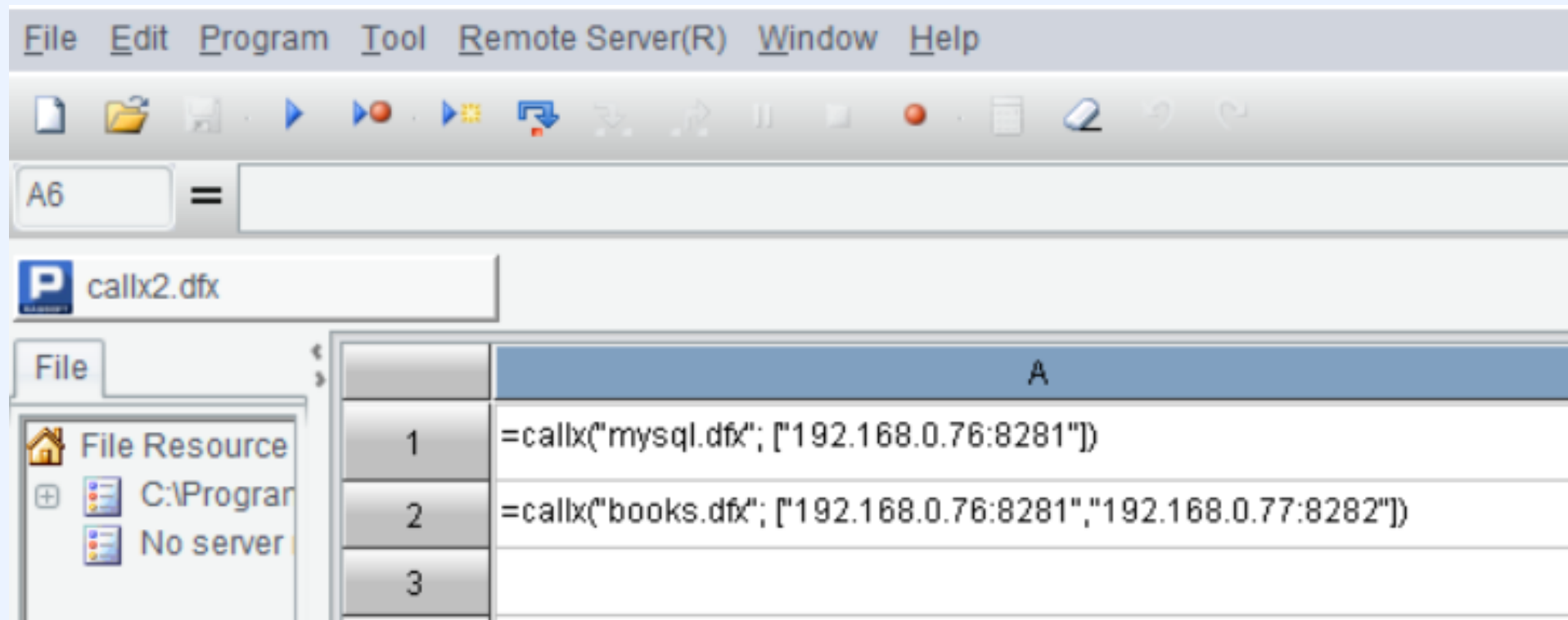
**RAQSOFT**



## 4、客户端使用



一旦用户登陆成功，Server端会启动docker容器。用户**进入IDE**后就可以操作，界面如下：



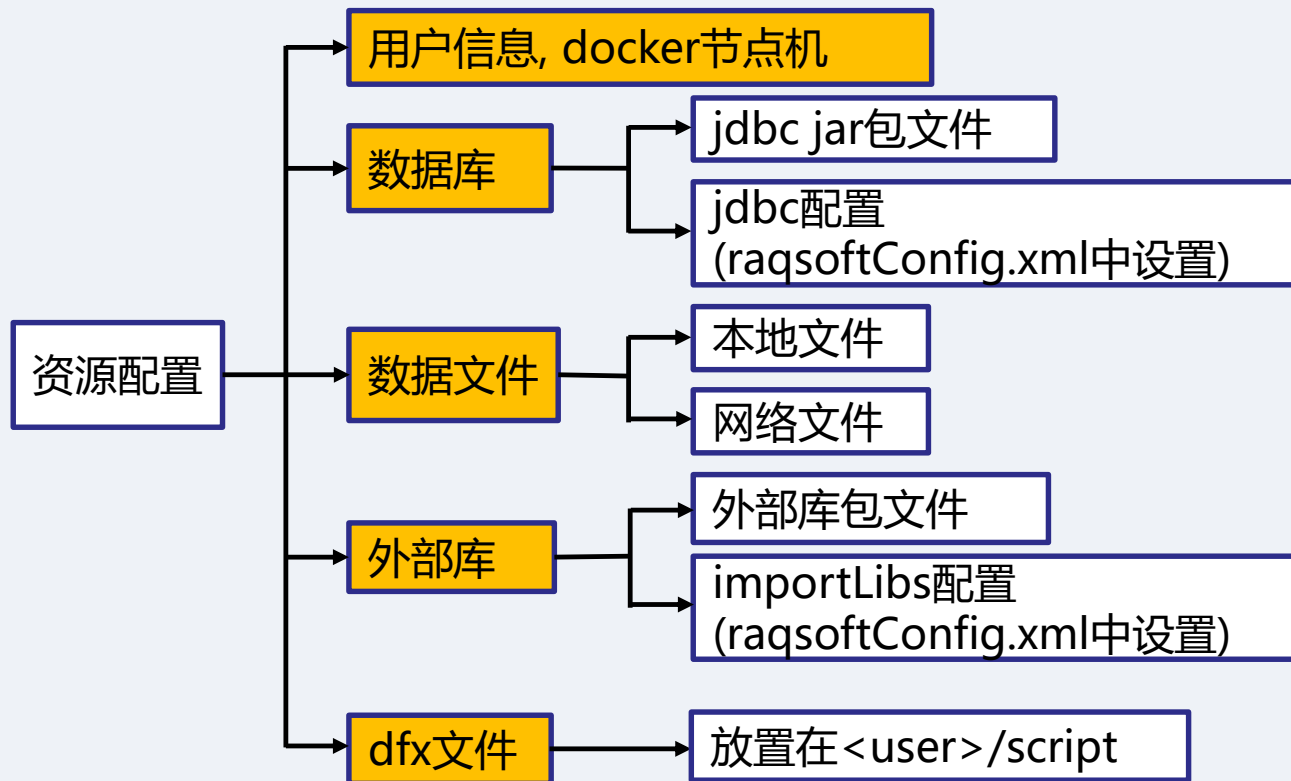
mysql.dfx, books.dfx文件放在服务器此用户的script目录下，如：/opt/app/aaa/script

## B、集算器资源配置



集算器镜像使用时，根据用户需求，**服务端**上可配置不同的资源，确保dfx脚本能正常运行。同时在**客户端**利用集算器接口callx调用远程的dfx脚本。（接口可参考<集算器函数参考>）

docker用户使用时，可参考下面资源配置图进行布署。



# C、文件数据访问



**需求:** 查询employee.txt中的STATE="New York", 参考下面配置列表

	Server	docker容器	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx脚本	script/Employee.dfx	script/Employee.dfx	loadEmp.dfx
数据	data/EMPLOYEE.txt	data/EMPLOYEE.txt	

## A1、Server端Employee.dfx脚本:

列出 docker 容器内容, 只为参考说明, 用户不需要操作它

	A	B
1	=file("/app/data/EMPLOYEE1.txt")	//读取文件,
2	=A1.import@t()	//导入数据
3	=A2.select(STATE==pState)	//pState过滤条件
4	return A3	//返回序表

# C、文件数据访问



## A2、Client端脚本loadEmp.dfx

	A	B
1	[192.168.0.76:8281]	//设置节点机
2	=callx("Employee.dfx ","New York";A1)	//调用Employee.dfx脚本
3	=A2.conj()	

### 返回结果:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
216	Brooke	Williams	F	New York	1978-12-03	2003-11-01	2	12000
385	Taylor	Wilson	F	New York	1980-05-15	2001-04-01	7	7000
221	Sarah	Davis	F	New York	1982-09-04	2007-03-01	4	5000
196	Lauren	Thomas	F	New York	1976-08-24	2009-08-01	2	12000
166	Emily	Moore	F	New York	1977-10-22	2006-01-01	3	7000
180	Abigail	Smith	F	New York	1972-09-19	2007-05-01	5	5000
297	Julia	Johnson	F	New York	1983-06-26	2000-09-01	7	5000

## 5、其它数据源



下面通过aaa用户的案例分析来说明如何使用的。

**测试环境:**

选项	Client	serverA	serverB
os	Win10	Ubuntu15	Ubuntu15
ip	192.168.0.186	192.168.0.76	192.168.0.77
port		8281	8282
docker管理	x	主服务器	从服务器
mysql	√	x	x
mongodb	√	x	x
<user>目录		/opt/app/aaa	/opt/app/aaa

## 5、其它数据源



### A、数据库的使用

**需求:** 用户需要使用mysql数据库, 镜像文件中没有安装mysql数据库及相关jar, 哪怎么办呢? docker可以访问外部的数据库, 不需要在docker中安装mysql, 参考下面配置列表。

	Server	docker容器	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx脚本	script/mysql.dfx	script/mysql.dfx	loadEmp.dfx
Jdbc配置	config/raqsoftConfig.xml		
Jdbc jar	/home/docker/share/jdbc/mysql-connector-java-5.1.31-bin.jar		

# 5、其它数据源



## A、数据库的使用

### A1、在raqsoftConfig.xml中配置mysql.

```
<DBList encryptLevel="0" >
  <DB name="mysql">
    <property name="url" value="jdbc:mysql://192.168.0.186:3306/docker"/>
    <property name="driver" value="com.mysql.jdbc.Driver"/>
    <property name="type" value="10"/>
    <property name="user" value="un"/>
    <property name="password" value="un1234"/>
    <property name="batchSize" value="0"/>
    <property name="autoConnect" value="false"/>
    <property name="useSchema" value="false"/>
    <property name="addTilde" value="false"/>
    <property name="needTransContent" value="false"/>
    <property name="needTransSentence" value="false"/>
    <property name="caseSentence" value="false"/>
  </DB>
</DBList>
```

# A、数据库的使用



**A2、** 检测/home/docker/share/jdbc目录下**mysql jdbc jar**文件是否存在，若不存在则管理员需要将mysql-connector-java-xxx-bin.jar放在jdbc目录下，启动docker时，会将jar文件复制到集算器jdbc依赖包下。

## **A3、mysql.dfx脚本**

	A	B
1	=connect("mysql")	
2	=A1.query("select pid,username,comment from t_user")	
3	=A1.close()	
4	return A5	

若是多台服务器，为了保证每个节点机正常运行，上述从A1到A3步骤都需要在每台服务器上进行同样的布署。

## **A4、Client端调用的loadMsql.dfx脚本：**

	A	B
1	=callx("mysql.dfx"; ["192.168.0.76:8281"])	



## B、外部库的使用



**需求：** 使用集算器的外部库，如Mongodb的使用，参考下面配置列表。

	Server	docker容器	Client
IP:PORT	192.168.0.76:8281	172.17.0.2:8281	192.168.0.186
<user>	/opt/app/aaa	/app	
dfx脚本	script/books.dfx	script/books.dfx	loadBooks.dfx
外部库配置	config/raqsoftConfig.xml		
mongo jar	/home/docker/share/extlib /MongoCli/*.jar	/share/MongoCli	

# B、外部库的使用



## D1、外部库包:

检测需要的外部库包是否存在, 若不存在则需要管理员将外部库包放在`/home/docker/share/extlib`目录下, 如MongoCli外部库, 则相关的jar包如下:

```
root@master:/home/docker/share/extlib# ls MongoCli
mongocli.jar  mongodb-driver-3.10.1.jar  mongo-java-driver-3.9.1.jar
```

## D2、配置文件

在raqsoftConfig.xml中增加MongoCli

```
<extLibsPath>/share/extlib</extLibsPath>
  <importLibs>
    <lib>MongoCli</lib>
  </importLibs>
```

**D3、用户需要重新登陆**, docker容器重启时, 再次启动集算器, 将加载Mongo外部库。

## D4、books.dfx脚本:

	A
1	=mongo_open("mongodb://192.168.0.186:27017/user")
2	=mongo_shell(A1,"books.find()").fetch()
3	=A2.groups(addr,book;count(book): Count)
4	=A3.groups(addr;sum(Count):Total)
5	=A3.join(addr,A4:addr,Total)

## B、外部库的使用



### D5、Client端调用的loadBooks.dfx脚本:

	A	B
1	=callx("books.dfx"; ["192.168.0.76:8281"])	

### 返回结果:

addr	book	Count	Total
address1	book1	3	4
address1	book5	1	4
address15	book1	1	1
address2	book1	2	3
address2	book5	1	3
address3	book9	1	1
address4	book3	1	1

## 6、集群的使用



**需求:** 查询employee.txt中的STATE="New York", 在多个docker容器中查询。

	ServerA	ServerB	Client
IP:PORT	192.168.0.76:8281	192.168.0.77:8282	192.168.0.186
<user>	/opt/app/aaa	/opt/app/aaa	
dfx脚本	script/Emp.dfx	script/Emp.dfx	loadEmp2.dfx
数据	data/EMPLOYEE.txt	data/EMPLOYEE.txt	
docker管理	主服务器	从服务器	
<Unit>	/home/docker/java/docker_users.xml	x	

### ServerA布署:

- A、查看docker\_users.xml中的<Unit>配置是否正确。
- B、将数据文件EMPLOYEE.txt放在用户目录/opt/app/aaa/data下。
- C、将dfx文件Emp.dfx放在用户目录/opt/app/aaa/script下。

**ServerB布署:** 参考ServerA中的步骤B、C进行同样布署。

## 6、集群的使用



### A、在Server端Emp.dfx脚本:

	A	B
1	=file("/app/data/EMPLOYEE1.txt")	//读取文件,
2	=A1.import@t(;pPart:pAll)	//分段导入数据
3	=A2.select(STATE==pState)	//pState过滤条件
4	return A3	//返回序表

### B、Client端脚本loadEmp2.dfx:

	A	B
1	[192.168.0.76:8281,192.168.0.77:8282]	//设置两个节点机
2	=callx("Emp.dfx","New York",[1,2],[2,2];A1)	//调用Emp.dfx脚本
3	=A2.conj()	//合并A2数据

A2将数据分成两段, 在节点机192.168.0.76:8281上查询前半段STATE="New York"的数据记录, 在节点机192.168.0.77:8282上查询后半段STATE="New York"的数据记录。

A3合并数据。

# 7、最后总结



作为docker管理员，导入集算器镜像后，需要设置docker管理服务，集群配置、集算器授权、外部库包及用户配置。

对于用户来说，根据自己的需求设置集算器配置、提供相关数据及计算使用的dfx脚本，然后在client端调用远程dfx脚本进行计算。

集算器集群，需要在多个服务器上配置，以便节点机协同作业，共同完成计算业务。