دانشکده مهندسی کامپیوتر

پروژه اول درس سیستم های توزیعی

نام دانشجو:

علیرضا بیدقی

مصطفی زمانی

استاد:

دکتر محسن شریفی

آذر ۱۳۹۲

کلاینت

کتابخانه و ابزارهای مورد استفاده:

- Glibtop برای گرفتن مشخصات CPU و Memory
- Pthread برای ساختن ریسه
- Iostat برای مشخصات دیسک سخت

معماری:

برای نمایش وضعیت سیستم از چهار متریک؛ میزان استفاده از پردازنده، میزان استفاده از حافظه ، نرخ خواندن از روی دیسک و نوشتن بر روی دیسک بر حسب بلاک بر ثانیه استفاده شده است. برای این منظور از چهار ریسمان مختلف که هر کدام وظیفه مشاهده یکی از متریک‌ها را بر عهده دارد، استفاده شده است. هر کدام از ریسمان‌ها در داخل خود یک ناحیه بحرانی دارند. . برای حفاظت از ناحیه بحرانی از mutex استفاده شده است.

برای نگه‌داری اطلاعات جمع‌آوری شده ، از ساختاری به نام SharedData استفاده‌شده. این ساختار دو بخش دارد. مقدار اندازه‌گیری شده و زمان اندازه‌گیری. هر کدام از ۴ ریسمان، یک نمونه از این ساختار را برای خوددارند.

یک ریسمان هم برای ارسال اطلاعات به سرور میانی وجود دارد. این ریسمان به صورت مداوم، اطلاعات دریافت شده از هر کدام از چهار ریسمان جمع کننده اطلاعات را به سرور می‌فرستد. فاصله زمانی ارسال اطلاعات هر متریک به سرور میانی، یک ثانیه است. در این ریسمان برای دسترسی به داده هر کدام از چهار ریسمان دیگر، چهار ناحیه بحرانی وجود دارد. برای حفاظت از ناحیه بحرانی از mutex استفاده شده است.

پردازه اصلی برنامه وظیفه دریافت اطلاعات از سرور را بر عهده دارد. در این برنامه تنها پیامی که از سرور می‌رسد، پیام خاتمه کار یک کلاینت است. اگر این پیام مربوط به همین کلاینت باشد، برنامه کلاینت بسته می‌شود.

توابع :

int main(int argc , char *argv[])

ورودی: نام کلاینت و آدرس IP سرور میانی

خروجی: ۰ یا کد خطای مربوط به بسته شدن ناگهانی برنامه

توضیح: تابع اصلی برنامه است و در آن پنج ریسمان دیگر برنامه مقداردهی اولیه شده و شروع به کار می‌کنند. این تابع درون یک حلقه بی‌نهایت ، منتظر پیغام Kill از طرف سرور است تا به کار برنامه خاتمه دهد.

static void * TimerRoutine(void * arg)

تابع اصلی رسیمان فرستنده اطلاعات به سرور است. در یک حلقه بینهایت، به فاصله یک ثانیه، اطلاعات مربوط به هر متریک را می‌فرستد.

void makeXML(char* name , char* mgName , char* content , char* time)

مقادیر لازم برای ساخت پیام را گرفته و پیام را به صورت xml ساخته و در یک متغیر سراسری قرار می‌دهد.

int isMyName(char *message)

پیغام Kill آمده از سرور را دریافت کرده نام آن را با نام کلاینت مقایسه می‌کند. در صورت همنام بودن، مقدار ۱ و در غیر این صورت مقدار ۰ را برمی‌گرداند.

void* getCpuUsage(void *arg)

با استفاده از تابع glibtop_get_cpu() در کتابخانه glibtop، در یک حلقه بینهایت، میزان استفاده از پردازنده را در یک بازه زمانی ۱۰۰ میلی ثانیه‌ای، اندازه‌گیری می‌کند.

void* getMemoryUsage(void *arg)

با استفاده از تابع glibtop_get_mem() در کتابخانه glibtop، در یک حلقه بینهایت، میزان حافظه استفاده شده را اندازه‌گیری می‌کند.

void* getFSRead(void *arg)

با استفاده از ابزار iostat ، نرخ خواندن از روی دیسک را بر حسب بلوک بر ثانیه اندازه‌گیری می‌کند.

void* getFSWrite(void *arg)

با استفاده از ابزار iostat ، نرخ نوشتن روی دیسک را بر حسب بلوک بر ثانیه اندازه‌گیری می‌کند.

سرور میانی

کتابخانه‌های مورد استفاده:

- Pthread: برای ساختن ریسه
- Socket:برای نوشتن سوکت

داده ساختار مورد استفاده:

- ClientData: در این داده ساختار اطلاعات مربوط به هر کلاینت ذخیره می‌شود.
  - Status: نشان می‌دهد که آیا مقدار این داده ساختار معتبر هست یا نه.
  - Name: نام کلاینت را نگه می‌دارد.
  - CPU: مقدار cpu را نگه می‌دارد.
  - MEM: مقدار mem را نگه می‌دارد.
  - FSR: مقدار سرعت خواندن از دیسک سخت را نگه می‌دارد.
  - FSW: مقدار سرعت نوشتن در دیسک سخت را نگه می دارد.
  - Socket: سوکتی که کلاینت از طریق آن به سرور میانی وصل شده است را نگه می‌دارد.
  - Timestamp: زمانی که این داده‌ها را ارسال کرده است.
- Arg_struct: برای پاس دادن دو متغیر به توابعی که ریسه آن‌ها را صدا می زند استفاده شده است.
  - Socket: سوکتی که کلاینت از طریق آن به سرور میانی وصل شده است.
  - Index: شماره اندیسی از آرایه‌ای که مشخصات این کلاینت در آن ذخیره می‌شود.

معماری:

ابتدا یک آرای به اندازه CLIENT_NUMBER از داده ساختار ClientData ساخته می‌شود تا اطلاعات کلاینت‌ها در آن ذخیره شود. سپس تابع initialize صدا زده می‌شود تا مقادیر اولیه آرایه فوق را مشخص نماید.بعد از آن سوکتی که قرار است به سرور اصلی وصل شود ساخته می‌شود و بعد از وصل شدن آن به سرور اصلی برای آن دو ریسه مختلف ساخته می‌شود یکی برای فرستادن داده‌ها به سرور اصلی و دیگری برای گرفتن داده از آن سرور. بعد از آن سوکتی که قرار است به عنوان welcome socket عمل کند ساخته می‌شود و به پورت ۲۴۲۵۱ وصل می‌شود و در یک حلقه بینهایت منتظر می‌ماند تا اگر یک کلاینت در خواست اتصال جدیدی داشت آن در خواست را قبول کرند. برای هر درخواست جدید (هر کلاینت جدید) دو ریسه مجزا ساخته می‌شود. در ریسه

connection_handler مدام به پورت گوش می‌دهد و داده‌های جدید کلایت را بر روی آرایه خود می‌نویسد برای آنکه در نوشتن و خواندن توسط ریسه‌های مختلف مشکلی پیش نیاید قبل از نوشتن در آرایه mutex مربوط به آن اندیس آرایه قفل می‌شود و بعد از نوشتن در آن قفل باز می‌شود. در ریسه دوم به نام outputEach5Second در هر ۵ ثانیه داده‌های آن کلاینت را بر روی سرور میانی نمایش می‌دهد. برای آنکه در هر لحظه فقط یک ریسه بتواند بنویسد یک mutex کلی در نظر گرفته شده است که هر ریسه خواست در خروجی چیزی بنویسد ابتدا آن را قفل کرده و بعد از نوشتن آن را باز می‌کند. در نهایت اگر دستور kill از طرف سرور اصلی بیاید توسط تابع killClients به تمامی کلاینت‌ها فرستاده می‌شود.

توابع:

در برنامه سرور میانی از توابع زیر استفاده شده است:

- void *connection_handler(void *args)
  - این تابع به ازای هر کلاینتی که به سرور میانی وصل می‌شود در یک ریسه مجزا اجرا می‌شود و داده‌های کلاینت را دریافت می‌کند و بر روی آرایه می‌نویسد.
- void* outputEach5Second(void *arg)
  - این تابع به ازای هر کلاینتی که به سرور میانی وصل می‌شود در یک حلقه تا زمانی که status آن یک است یعنی داده‌ها معتبر هستند هر ۵ ثانیه اطلاعات آن کلاینت را بر روی سرور میانی نمایش می‌دهد.
  - اگر از آخرین باری که داده برای سرور میانی آمده باشد بیش از ۵ ثانیه گذشته باشد فرض می‌کند که کلاینت kill شده است و داده‌های آن را نامعتبر می‌سازد.
- int findFirstIndex()
  - اولین جای خالی را در آرایه‌ای که برای ذخیره‌سازی اطلاعات کلاینت‌ها در نظر گرفته شده است بر می‌گرداند. یعنی اولین جایی که مقدار status آن برابر صفر است.
- void *sendToMainServer(void *arg)
  - در یک حلقه بینهایت شروع می‌کند و به ترتیب اطلاعات کلاینت‌ها را برای سرور اصلی می‌فرستد.
- void *recieveFromMainServer(void *arg)
  - در یک حلقه بی‌نهایت منتظر داده از سرور اصلی می‌ماند.
  - اگر داده آمده دستور kill بود تابع مربوط به آن را صدا می زند.
- void initialize()
  - برای مقداردهی اولیه آرایه‌ای که داده‌های کلاینت‌ها در آن ذخیره می‌شود.
- void readXML(char* XMLmsg,char* name , char* mgName , char* content , char* mytime)
  - برای آنکه بتواند از پیامی که از طرف کلاینت آمده و به صورت XML هست را داده‌های مورد نیاز را استخراج کند.
  - آرگومان اول ورودی تابع و بقیه آن‌ها خروجی تابع هستند.
- int updateTimestamp(char* newTimestamp, int index)

- برای بررسی آنکه پیام رسیده از طرف کلاینت آخرین پیام هست یا نه از این تابع استفاده می‌شود. در زمان آن از زمان آخرین پیام آمده بیشتر بود که آن را به‌روز می‌کند در غیر این صورت داده را رد می‌کند.

- void makeXML(char* XMLmsg ,char* name , char* cpu , char* mem , char* fsw ,char* fsr ,char* mytime)
  - برای ساختن پیامی که قرار است به ازای هر کلاینت به صورت XML باشد و به سرور اصلی فرستاده شود از این تابع استفاده می‌شود.
  - آرگومان اول خروجی تابع و بقیه آرگومان‌ها ورودی آن می‌باشند.
- void killClients(char *message)
  - برای ارسال پیام kill به کلاینت‌ها از این تابع استفاده می‌شود.
  - اگر نام این کلاینت در داخل آرایه بود برای همه کلاینت‌ها این پیام را می‌فرستد.
- int isClientName(char *message,char* ClientName)
  - برای بررسی پیام آمده با یک اسم خاص که در واقع این اسم همان نام کلاینت‌ها می‌باشد.

سرور اصلی

کتابخانه‌های مورد استفاده:

- Pthread: برای ساختن ریسه
- Socket:برای نوشتن سوکت

داده ساختار مورد استفاده:

- ClientData: در این داده ساختار اطلاعات مربوط به هر کلاینت ذخیره می‌شود.
  - Status: نشان می‌دهد که آیا مقدار این داده ساختار معتبر هست یا نه.
  - isFirstTime: اولین باری هست که مقداردهی می‌شود یا خیر.
  - midServer_socket: برای نگه‌داری سوکت سرور میانی که داده‌های این کلاینت را می‌فرستد.
  - Name: نام کلاینت را نگه می‌دارد.
  - CPU: مقدار cpu را نگه می‌دارد.
  - MEM: مقدار mem را نگه می‌دارد.
  - FSR: مقدار سرعت خواندن از دیسک سخت را نگه می‌دارد.
  - FSW: مقدار سرعت نوشتن در دیسک سخت را نگه می دارد.
  - Timestamp: زمانی که این داده‌ها را ارسال کرده است.
  - firstTimestamp: زمان اولین ارسال.
  - serverTime: زمان سرور در لحظه به‌روزرسانی اطلاعات این کلاینت.
  - countCpu: تعداد ارسال‌های مربوط به مقدارCPU برای میانگین‌گیری.
  - countMem: تعداد ارسال‌های مربوط به مقدار mem برای میانگین‌گیری.
  - countFsr: تعداد ارسال‌های مربوط به مقدارfsr برای میانگین‌گیری.
  - countFsw: تعداد ارسال‌های مربوط به مقدارfsw برای میانگین‌گیری.
  - avgCpu: میانگین cpu
  - avgMem: میانگین mem
  - avgFsr: میانگین fsr
  - avgFsw: میانگین fsw
- Arg_struct: برای پاس دادن دو متغیر به توابعی که ریسه آن‌ها را صدا می زند استفاده شده است.
  - Socket: سوکتی که کلاینت از طریق آن به سرور میانی وصل شده است.
  - Index: شماره اندیسی از آرایه‌ای که مشخصات این کلاینت در آن ذخیره می‌شود.

معماری:

در ابتدا یک سوکت ساخته می‌شود و به پورت 8852 وصل می‌شود. سپس برای آن یک ریسه جدا تعریف می‌شود تا در آن ریسه در یک حلقه بینهایت منتظر بماند تا اگر سرور میانی جدید آمد آن سرور میانی را قبول کند. در ریسه‌ای که منتظر یک سرور میانی جدید است اگر سرور میانی جدیدی آمد برای آن یک ریسه جدید می‌سازد و آن ریسه جدید مسئول گرفتن پیام از سرور میانی و نوشتن داده‌های آن در آرایه مربوط به کلاینت‌ها است. بعد از آن برنامه در یک حلقه بینهایت منتظر گرفتن دستور از کاربر می‌شود و با گرفتن هر یک از دستورات تابع مربوط به آن را صدا می زند. در انتها برای بررسی آنکه کلاینت‌هایی که داده‌های آن بروز نشده است یک ریسه جداگانه در نظر گرفته شده است که در یک حلقه بینهایت نگاه می‌کند اگر از آخرین به‌روزرسانی آن بیش از ۳ ثانیه گذشته است داده آن را نامعتبر می‌کند.

توابع:

در برنامه سرور از توابع زیر استفاده شده است:

- void *listenForAcceptMidServer(void *arg)
  - در یک حلقه بینهایت منتظر آمدن سرور میانی است تا اگر سرور میانی آمد برای آن یک ریسه جدا بسازد.
- void sendKillMessage(char* clientName,int socket)
  - فرستادن پیام kill به سرورهای میانی.
- void *connection_handler(void *args)
  - به ازای هر سرور میانی در یک حلقه بینهایت پیام‌ها را از سرور اصلی می‌گیرد.
  - به‌روزرسانی داده‌های مربوط به کلاینت.
  - میانگین‌گیری از داده‌های جدیدی که آمده با داده‌های قبلی.
- int findFirstIndex()
  - پیدا کردن اولین جای خالی در آرایه‌ای که داده‌های کلاینت‌ها را نگه‌داری می‌کند.
- void readXML(char* XMLmsg,char* name , char* cpu, char* mem, char* fsr, char* fsw, char* time)
  - استخراج پیام آمده از سرور میانی که به صورت XML است و تبدیل آن به داده‌های قابل‌استفاده.
  - آرگومان اول ورودی تابع و بقیه آن‌ها خروجی هستند.
- void initialize()
  - مقداردهی اولیه به آرایه‌ای که داده‌های کلاینت‌ها در آن نگه داشته می‌شوند.
- void show()
  - نشان دادن نام کلاینت‌ها در سرور اصلی بر روی مانیتور.
- void now(int index)
  - نشان دادن آخرین وضعیت یک کلاینت خاص.

- void avg(int index)
  - نشان دادن میانگین یک کلاینت خاص.
- void nowAll()
  - نمایش آخرین وضعیت تمامی کلاینت‌ها.
- void avgAll()
  - نمایش میانگین وضیعت تمامی کلاینت‌ها.
- void mykill(int i)
  - نامعتبر کردن داده مربوط به یک کلاینت خاص بعد از اجرای دستور Kill از طرف کاربر.
- int findIndexByName(char* name)
  - پیدا کردن اندیس آرایه که کلاینت‌ها در آن ذخیره‌شده‌اند توسط نام آن کلاینت.
- void* checkForKill(void* arg)
  - بررسی می‌کند که اگر از آخرین به‌روزرسانی یک کلاینت خاص بیش از ۵ ثانیه گذشته است داده‌های مربوط به آن کلاینت را نامعتبر می‌کند.