Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <iostream>

#include <cstdio>

#include <cstring>

#include <unordered\_map>

#include <omp.h>

#include <vector>

using namespace std;

const char\* pathToFile = ".log";

void readLogFile()

{

FILE\* logFile = fopen(pathToFile, "r");

if (!logFile)

{

cout << "Open file error" << endl;

return;

}

char line[1024];

unordered\_map<string, int> methods = { {"GET", 0}, {"POST", 0}, {"PUT", 0}, {"DELETE", 0}, {"HEAD", 0} };

vector<string> logLines;

while (fgets(line, sizeof(line), logFile))

{

logLines.push\_back(line);

}

fclose(logFile);

double time = omp\_get\_wtime();

for (size\_t i = 0; i < logLines.size(); ++i)

{

const char\* method = strstr(logLines[i].c\_str(), "GET") ? "GET"

: strstr(logLines[i].c\_str(), "POST") ? "POST"

: strstr(logLines[i].c\_str(), "PUT") ? "PUT"

: strstr(logLines[i].c\_str(), "DELETE") ? "DELETE"

: strstr(logLines[i].c\_str(), "HEAD") ? "HEAD"

: nullptr;

if (method)

{

methods[method]++;

}

}

cout << endl << 1 << " thread results:" << endl;

for (const auto& pair : methods)

{

cout << pair.first << ": " << pair.second << endl;

}

cout << "Time: " << (omp\_get\_wtime() - time)\*1000 << " ms" << endl;

}

void readLogFileParallel(int nThreads)

{

FILE\* logFile = fopen(pathToFile, "r");

if (!logFile)

{

cout << "Open file error" << endl;

return;

}

char line[1024];

unordered\_map<string, int> methods = {{"GET", 0}, {"POST", 0}, {"PUT", 0}, {"DELETE", 0}, {"HEAD", 0} };

vector<string> logLines;

while (fgets(line, sizeof(line), logFile))

{

logLines.push\_back(line);

}

fclose(logFile);

double time = omp\_get\_wtime();

#pragma omp parallel num\_threads(nThreads)

{

unordered\_map<string, int> localMethods = {{"GET", 0},

{"POST", 0},

{"PUT", 0},

{"DELETE", 0},

{"HEAD", 0} };

#pragma omp for

for (size\_t i = 0; i < logLines.size(); ++i)

{

const char\* method = strstr(logLines[i].c\_str(), "GET") ? "GET"

: strstr(logLines[i].c\_str(), "POST") ? "POST"

: strstr(logLines[i].c\_str(), "PUT") ? "PUT"

: strstr(logLines[i].c\_str(), "DELETE") ? "DELETE"

: strstr(logLines[i].c\_str(), "HEAD") ? "HEAD"

: nullptr;

if (method)

{

localMethods[method]++;

}

}

#pragma omp critical

{

for (const auto& pair : localMethods)

{

methods[pair.first] += pair.second;

}

}

}

cout << endl << nThreads << " threads results:" << endl;

for (const auto& pair : methods)

{

cout << pair.first << ": " << pair.second << endl;

}

cout << "Time: " << (omp\_get\_wtime() - time) \* 1000 << " ms" << endl;

}

int main()

{

readLogFile();

readLogFileParallel(2);

readLogFileParallel(4);

readLogFileParallel(8);

readLogFileParallel(12);

return 0;

}