Разработка high-performance web server бенчмарка httpress

```
// Block signals for all threads
sigset_t set;
sigemptyset(&set);
sigaddset(&set, SIGTERM);
sigaddset(&set, SIGPIPE);
sigaddset(&set, SIGINT);
sigaddset(&set, SIGQUIT);
sigaddset(&set, SIGHUP);
if (pthread_sigmask(SIG_BLOCK, &set, NULL)) {
    nxweb_log_error(NULL, "can't set pthread_sigmask");
    exit(EXIT_FAILURE);
}

if (i_opt_ip4_addr) {
    const char *prt;
    int port;
    struct sockaddr_in *addr;
    struct addrinfo ainf;

    ainf.ai_family = AF_INET;
    ainf.ai_flags = 0;
    ainf.ai_protocol = 0;
    ainf.ai_canonname = NULL;
    ainf.ai_socktype = SOCK_STREAM;
    ainf.ai_addrlen = sizeof(struct sockaddr_in);
                                        1976,47-53    88%
```

Технологии: разработка на Си под Linux

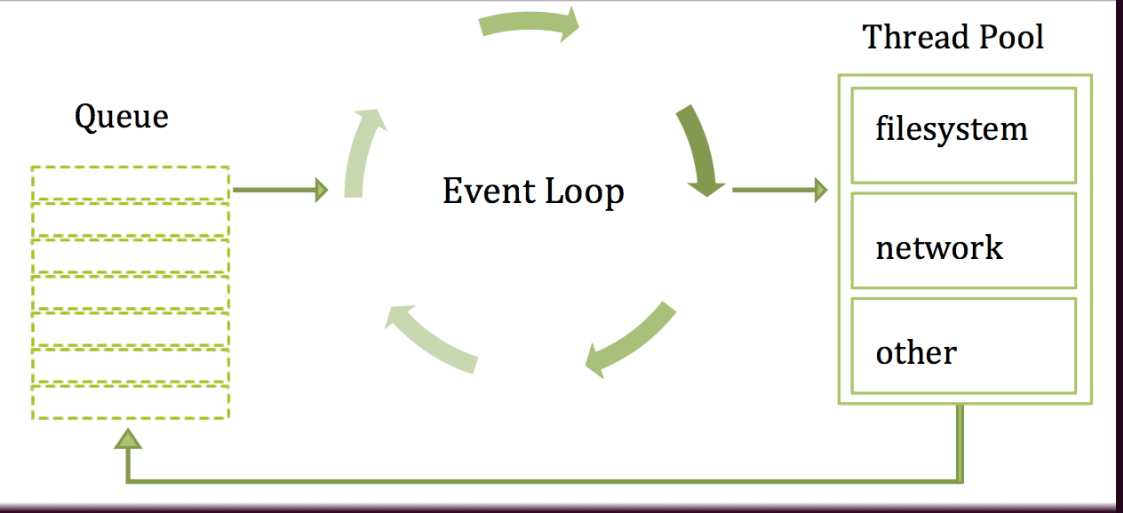
HTTP

```
}
else {
    for (int idx = 0; idx < common_config.tot_domains_number; ++idx)
        if (resolve_host(&config[idx].saddr, config[idx].uri_host, idx)) {
            nxweb_log_error(NULL, "can't resolve host %s", config[idx].uri_host);
            exit(EXIT_FAILURE);
        }
}

for (int idx = 0; idx < common_config.tot_domains_number; ++idx) {
    snprintf(config[idx].request_data, sizeof(config[idx].request_data),
            "GET %s HTTP/1.1\r\n"
            "Host: %s\r\n"
            "Connection: %s\r\n",
            config[idx].uri_path,
            config[idx].uri_host,
            common_config.keep_alive ? "keep-alive" : "close"
            );
    config[idx].request_length = strlen(config[idx].request_data);
}

thread_config** threads = calloc(common_config.num_threads, sizeof(thread_config*));
if (!threads)
    nxweb_die("can't allocate thread pool");

ev_tstamp ts_start = ev_time();
```



Event loop, libev, low cpu consumption

```c
static inline int setup_socket(int fd)
{
    int flags = fcntl(fd, F_GETFL);
    if (flags < 0)
        return flags;
    if (fcntl(fd, F_SETFL, flags |= O_NONBLOCK) < 0)
        return -1;

    int nodelay = 1;
    if (setsockopt(fd, IPPROTO_TCP, TCP_NODELAY, &nodelay,
        return -1;
```

```c
typedef int atomic_val_t;
typedef struct { volatile atomic_val_t val; } atomic_t;

atomic_t reqs_in_flight;    // number of sent HTTP packets

static inline atomic_val_t atomic_get(atomic_t *p)
{
    return p->val;
}

static inline atomic_val_t atomic_inc(atomic_t *p)
{
    return __sync_add_and_fetch(&p->val, 1);
}
```

TCP,

Concurrency

Задачи

- Реализация HTTP GET, POST, PUT, DELETE запросов с параметрами
- Реализация функционала для создания сценариев тестирования

Спасибо

eabatalov89@gmail.com