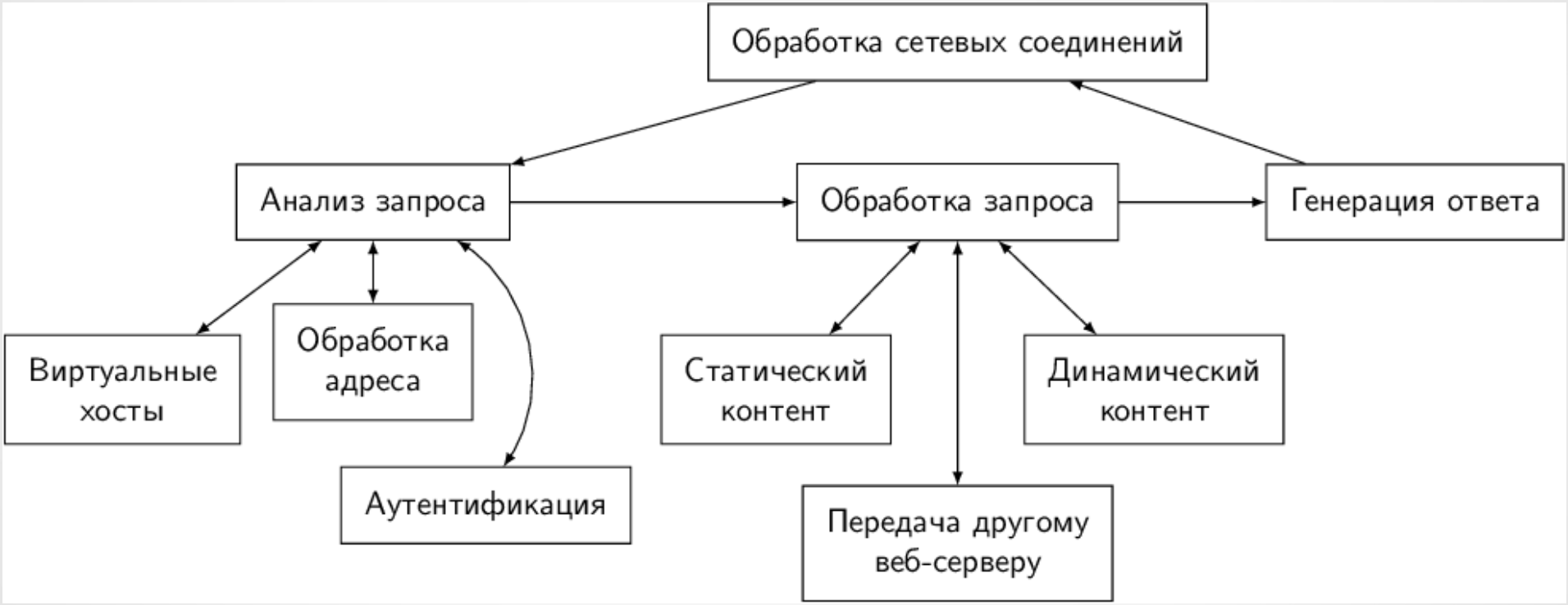
Архитектура Веб-сервера

Кулаков Кирилл Александрович

Основные функции Веб-сервера

- Управление соединением
- Прием и обработка запроса
- Разделение доступа к нескольким обслуживаемым наборам ресурсов
- Отдача статического содержимого
- Взаимодействие с приложениями для получения и дальнейшей отдачи клиенту динамического содержимого

Высокоуровневая архитектура Веб-сервера



Модели обработки соединений

- forking server
- pre-forking server
- worker threads
- event-driven

forking server

- Основной процесс в цикле ожидает новых соединений на системном вызове accept (если сокетов более одного обычно применяют select, poll, epoll).
- После получения очередного клиентского сокета основной процесс форкается (fork) в результате чего файловый дескриптор наследуется (копируется в таблицу дескрипторов дочернего процесса с увеличением счетчика ссылок).
- Затем родительский процесс закрывает (close) свою копию дескриптора и возвращается к ожиданию нового соединения.
- Дочерний процесс продолжает обслуживание клиента в отдельном изолированном потоке исполнения.

pre-forking server

- родительский процесс в самом начале создает один или несколько серверных сокетов (те, для которых сделали listen) и расфоркивается (fork) столько раз, сколько рабочих процессов в пуле мы хотим иметь.
- Далее он может ничего не делать, а просто ждать завершения работы дочерних процессов (при этом рестартовать их) и сигналов снаружи.
- Дочерние процессы все вместе в цикле висят ожидая входящего соединения на серверном сокете (accept).
- Как приходит новый клиент только один процесс разблокируется и получит клиентский сокет, остальные будут продолжать ждать.

worker threads

- Worker сочетает в себе мульти-процессовый (multi-process) и мульти-поточный (multi-threaded) сервер. Использование потоков (threads) позволяет обслуживать больше запросов при меньших аппаратных ресурсах чем просто мульти-процессовый сервер (prefork).
- родительский процесс отвечает за запуск дочерних процессов;
- каждый дочерний процесс создает фиксированное количество потоков, установленных директивой ThreadsPerChild и один слушающий поток (listener thread), который принимает новые соединения и передает их рабочим потокам;

event-driven

- Event позволяет обслуживать большое количество одновременных соединений путем передачи части работы по обработке запроса отдельным потокам (listeners threads).
- Модель призвана исправить проблему "keep alive" в HTTP.
- После того, как клиент выполнит первый запрос, он может оставить соединение открытым, отправляя будущие запросы используя тот же самый сокет избегая накладных расходов при установлении нового TCP-соединения.
- В случае prefork и worker сохраняется дочерний процесс / поток, ожидающий данных от клиента, что приводит к неэффективному расходу ресурсов.
- Чтобы решить эту проблему, event использует выделенный поток (listener thread) для каждого процесса для обработки сокетов в состоянии Listening, Keep Alive и сокетов, для которых осталось только передать данные клиенту.
- Таким образом рабочий поток получает уже готовый запрос и освобождается сразу после его обработки.

Способы генерации динамического содержимого

- Шаблоны
  - SSI (Server Side Includes)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Server Side Includes Example</title>
    </head>
    <body>
        <!--#include file="menu.html"-->
        <p>
            Some content
        </p>
    </body>
</html>
```

Способы генерации динамического содержимого

- Шаблоны
  - ColdFusion

```
<cfquery name="nameofquery" datasource="odbc_connection" username="simple" password="enough">
   SELECT * FROM table
   WHERE field = 'whateveryouaresearchingfor'
</cfquery>
<cfoutput query="nameofquery">
 #field_from_query#
 <!--- Выше находится вывод переменной, а это — просто комментарий. --->
</cfoutput>
```

Способы генерации динамического содержимого

- Взаимодействие со сторонним (отдельным от веб-сервера) приложением
  - Приложение не зависит от веб-сервера: CGI (Common Gateway Interface), FastCGI, SCGI.
  - Приложение зависит от сервера: Native APIs (ISAPI, NSAPI, Apache API), WSGI, сервлеты.
  - Смешанный подход — модуль веб-сервера запускает интерпретатор некоторго динамического языка, на котором написано приложение.

Способы генерации динамического содержимого

- Обратное проксирование — приложение уже предоставляет HTTP-интерфейс, серверу необходимо только перенаправлять полученные запросы к нему.

Сервер приложений

- Сервер приложений — программная среда, предоставляющая средства для создания и выполнения веб-приложений, а также обслуживающая запросы к ним.
- Сервер приложений может обслуживать запросы по протоколу HTTP, а может только предоставлять интерфейсы для взаимодействия со внешним веб-сервером.
- В задачи веб-сервера общего назначения, как правило, входят функции кеширования, обслуживания запросов к статическим ресурсам, эффективной обработке входящих запросов.

Комбинированная архитектура веб приложения