avito.tech | MTC

avito.tech MTC

ASYNC / AWAIT

or how to XXX

Maxim **Surkov**
iOS Developer at MTS

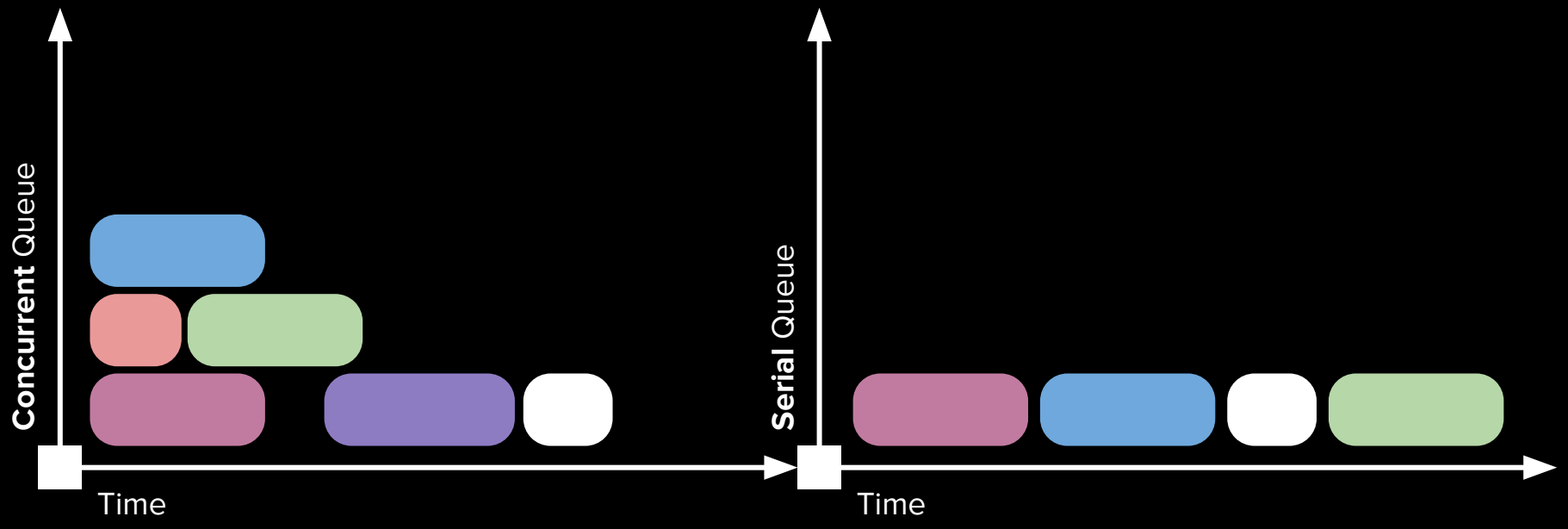
План

avito.tech МТС

- Как все работает сейчас
- Какое ключевое отличие работы функции через async/await от обычной
- Различные Tasks
- Actors как инструмент
- Переход на structured concurrency
- Итоги

Основы (Lays & Pringles)

avito.tech MTC



R8s4TqKp9Yw2XuVz3F6G

GCD - Swift multithreading foundation

avito.tech MTC

```
func doSome() {
 let group = DispatchGroup()
 group.enter()

 viewModel.fetchWeatherData
 viewModel.fetchNewsData

 group.leave()
}
```

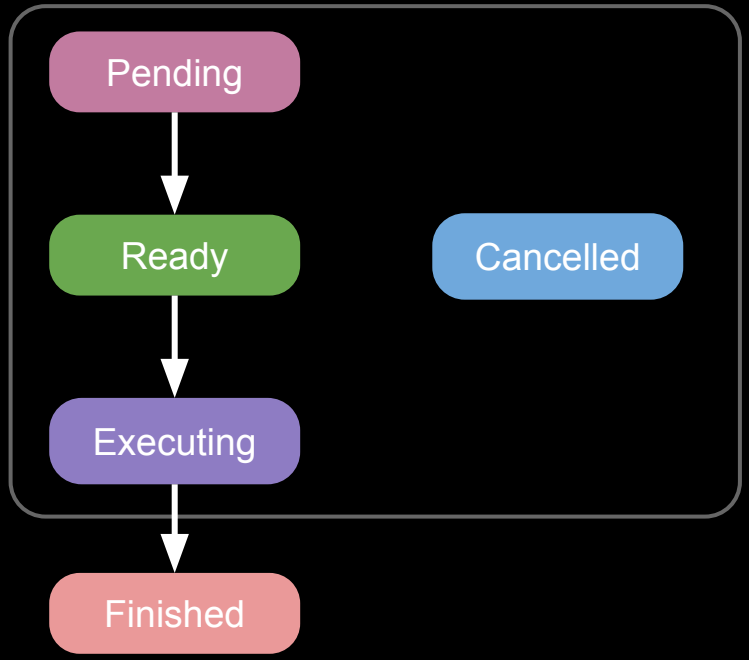
```
func doSome() {
 DispatchQueue.main.async {
  // some code
 }
}
```

QOS

- BACKGROUND
- DEFAULT
- UNSPECIFIED
- USERINITIATED
- USERINTERACTIVE
- UTILITY

OperationQueue

avito.tech MTC




```
let simplestOperation = {
        print("SimplestOperation start")
        print("SimplestOperation finished")
    }
let queue = OperationQueue()
queue.addOperation(simplestOperation)
```

```
open class Operation: NSObject {
  open func start()
  open func main()
  open var isCancelled: Bool { get }
  open func cancel()
  open var isExecuting: Bool { get }
  open var isFinished: Bool { get }
  open var isAsynchronous: Bool { get }
  open var isReady: Bool { get }

  open var completionBlock: (() -> Void)?
  open var qualityOfService: QualityOfservice
  ...
}
```

pthreads

avito.tech MTC

pthread

- execution
- creation
  - pthread_t
  - pthread_attr_t
  - pthread_create
- synchronization
  - pthread_join
  - mutex
  - conditional

POSIX - треды, обычно называемые pthreads, являются моделью выполнения, которая существует независимо от языка программирования, а также параллельной моделью выполнения. Они позволяют программе контролировать несколько различных потоков работы, которые перекрываются по времени.

R8s4TqKp9Yw2XuVz3F6G

Как все работает внутри?

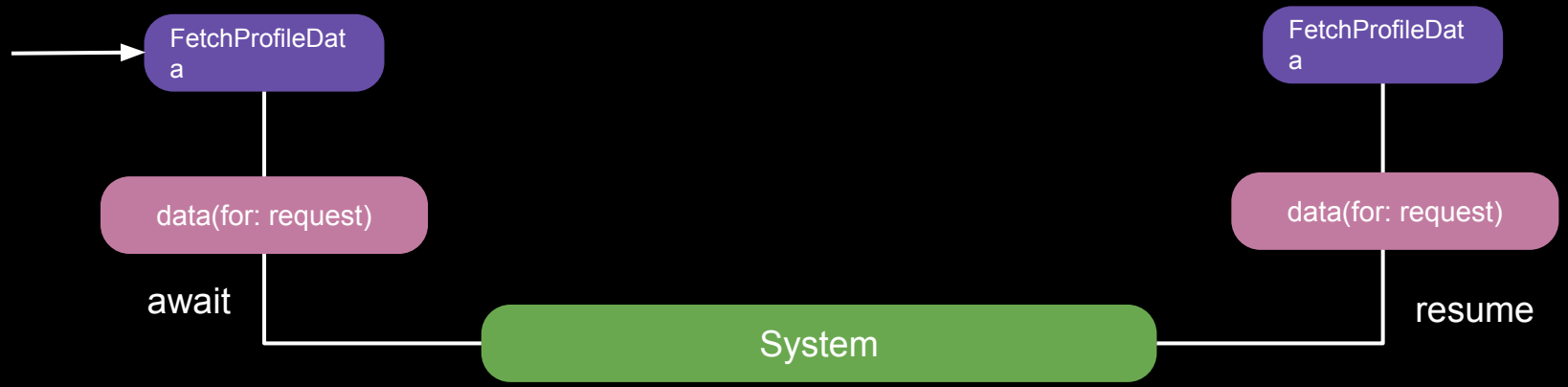
How it works now?

avito.tech MTC


```
func fetchProfileData(id: String) -> ProfileData {
    return req
}
```

How it works in async/await?

avito.tech MTC



Async / Await

avito.tech MTC

```
func markAsViewed(
    cameraId: String,
    clipIds: Set<String>
) async throws {
    try await convert { key, closure in
        RequestManager.archieveMarkViewed(
          cameraPublicId: cameraId,
          clipIds: clipIds,
          contextKey: key,
          completion: closure
        )
    }
)
```

```
extension UIImage {
    var thumbNail: UIImage? {
        get async {
            let size = CGSize(width: 40, height: 40)
            return await self.byPreparingThumbnail(ofSize: size)
        }
    }
}
```

Реалии разработки

avito.tech MTC

```
func processImageData2c(completionBlock: (Result<Image, Error>) -> Void) {
    loadWebResource("dataprofile.txt") { dataResourceResult in
        switch dataResourceResult {
        case .success(let dataResource):
          loadWebResource("imagedata.dat") { imageResourceResult in
              switch imageResourceResult {
              case .success(let imageResource):
                decodeImage(dataResource, imageResource) { imageTmpResult
in
                    switch imageTmpResult {
                    case .success(let imageTmp):
                      dewarpAndCleanupImage(imageTmp) { imageResult in
                          completionBlock(imageResult)
                      }
                    case .failure(let error):
                      completionBlock(.failure(error))
                    }
                case .failure(let error):
                    completionBlock(.failure(error))
              }
            }
            case .failure(let error):
                completionBlock(.failure(error))
          }
      }
}
```

R8s4TqKp9Yw2XuVz3F6G

Реалии разработки

avito.tech MTC

Leaks

Error handling

Conditional execution

```
func processImageData2c(completionBlock: (Result<Image, Error>) -> Void) {
    loadWebResource("dataprofile.txt") { dataResourceResult in
        switch dataResourceResult {
        case .success(let dataResource):
          loadWebResource("imagedata.dat") { imageResourceResult in
              switch imageResourceResult {
              case .success(let imageResource):
               decodeImage(dataResource, imageResource) { imageTmpResult
in
                    switch imageTmpResult {
                    case .success(let imageTmp):
                      dewarpAndCleanupImage(imageTmp) { imageResult in
                          completionBlock(imageResult)
                      }
                     case .failure(let error):
                      completionBlock(.failure(error))
                    }
               case .failure(let error):
                    completionBlock(.failure(error))
              }
          }
          case .failure(let error):
               completionBlock(.failure(error))
        }
    }
}
```

Реалии разработки

avito.tech MTC

Leaks

Error handling

Conditional execution

```
func processImageData2c(completionBlock: (Result<Image, Error>) -> Void) {
    loadWebResource("dataprofile.txt") { dataResourceResult in
        switch dataResourceResult {
        case .success(let dataResource):
          loadWebResource("imagedata.dat") { imageResourceResult in
              switch imageResourceResult {
              case .success(let imageResource):
               decodeImage(dataResource, imageResource) { imageTmpResult
in
                    switch imageTmpResult {
                    case .success(let imageTmp):
                      dewarpAndCleanupImage(imageTmp) { imageResult in
                          completionBlock(imageResult)
                      }
                     case .failure(let error):
                      completionBlock(.failure(error))
                    }
               case .failure(let error):
                    completionBlock(.failure(error))
              }
          }
          case .failure(let error):
               completionBlock(.failure(error))
         }
      }
}
```

Реалии разработки

avito.tech MTC

Leaks
Error handling
Conditional execution

```
func processImageData2c(completionBlock: (Result<Image, Error>) -> Void) {
    loadWebResource("dataprofile.txt") { dataResourceResult in
        switch dataResourceResult {
        case .success(let dataResource):
          loadWebResource("imagedata.dat") { imageResourceResult in
              switch imageResourceResult {
              case .success(let imageResource):
               decodeImage(dataResource, imageResource) { imageTmpResult
in
                    switch imageTmpResult {
                    case .success(let imageTmp):
                      dewarpAndCleanupImage(imageTmp) { imageResult in
                          completionBlock(imageResult)
                      }
                     case .failure(let error):
                      completionBlock(.failure(error))
                    }
               case .failure(let error):
                    completionBlock(.failure(error))
              }
          }
          case .failure(let error):
               completionBlock(.failure(error))
         }
      }
}
```

Новый подход

avito.tech MTC

Перформанс для асинхронного кода

Фундамент для будущих функций параллелизма, таких как task priority и их отмена.

Единообразный процесс отладки, профилирования и изучения кода.

```
func loadWebResource(_ path: String) async throws -> Resource
    func decodeImage(_ r1: Resource, _ r2: Resource) async throws -> Image
    func dewarpAndCleanupImage(_ i : Image) async throws -> Image
    func processImageData() async throws -> Image {
        let dataResource  = try await loadWebResource("dataprofile.txt")
        let imageResource = try await loadWebResource("imagedata.dat")
        let imageTmp      = try await decodeImage(dataResource,
imageResource)
        let imageResult   = try await dewarpAndCleanupImage(imageTmp)
        return imageResult
    }
}
```

Easy convertibility

avito.tech MTC

Finder Sun 1:44 PM

```
func data
    guard
        c
        r
    }

    let r

    let t
        E
        i

        }
        e

        }
    }
    task.resume()
}
```

- Jump to Definition
- Show Code Actions
- Show Quick Help
- Create Column Breakpoint
- Continue to Here
- Refactor
- Find
- Navigate
- Show Last Change for Line
- Create Code Snippet...
- Add Pull Request Discussion to Current Line
- Cut
- Copy
- Copy File and Line
- Copy Symbol Name
- Copy Qualified Symbol Name
- Paste
- Services

```
aping (Result<T, Error>) -> Void) {

ata?, response: URLResponse?, error:

T.self, from: data)
```

Easy convertibility

avito.tech

Finder

Sun 1:44 PM

```
func dataTask<T: Codable>(_ path: String) async throws -> T {
    guard let url = makeURL(path: path) else {
        throw InternalError.unknownPath
        return
    }

    let request = URLRequest(url: url)

    return try await withCheckedThrowingContinuation { continuation in
        let task = session.dataTask(with: request) { (data, response, error) in
            if let error = error {
                DispatchQueue.main.async {
                    continuation.resume(with: .failure(error))
                }
            }
            else if let data = data {
                do {
                    let result = try self.jsonDecoder.decode(T.self, from: data)
                    DispatchQueue.main.async {
                        continuation.resume(with: .success(result))
                    }
                }
                catch let error {
                    DispatchQueue.main.async {
                        continuation.resume(with: .failure(error))
                    }
                }
            }
        }
        task.resume()
```

R8s4TqKp9Yw2XuVz

Tasks

Tasks

avito.tech MTC

Task создает новый асинхронный контекст для параллельного выполнения кода.

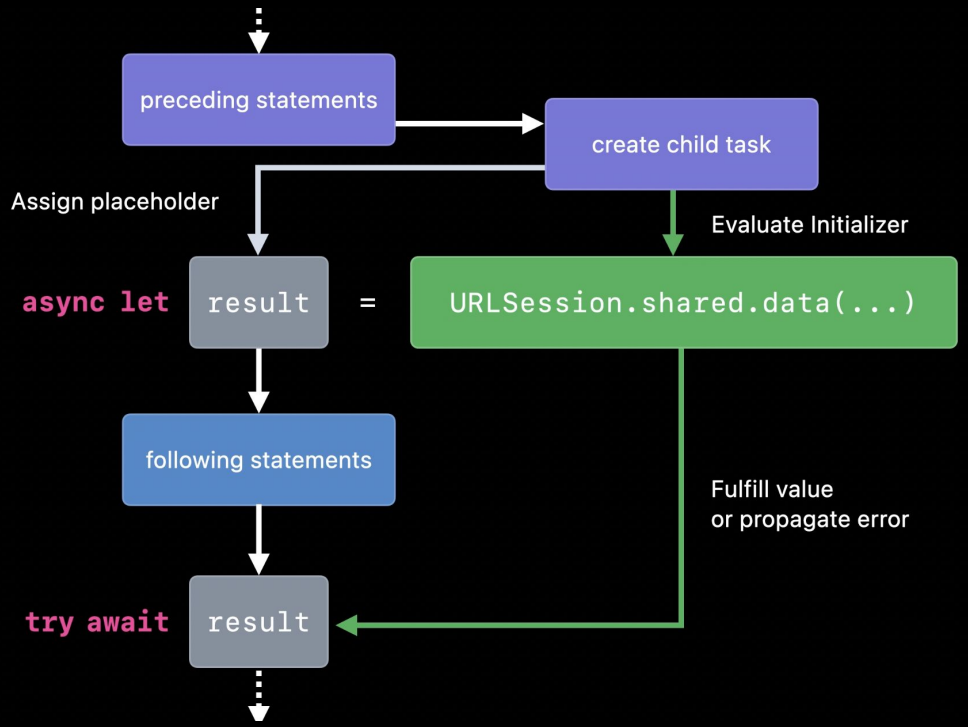
Swift проверяет использование вами task-ов, чтобы помочь предотвратить ошибки параллелизма.

При вызове асинхронной функции в Swift task явно не создается.

Async let Tasks

avito.tech MTC

```
async let thing = avito()
//some stuff
doAvito(await thing)
```

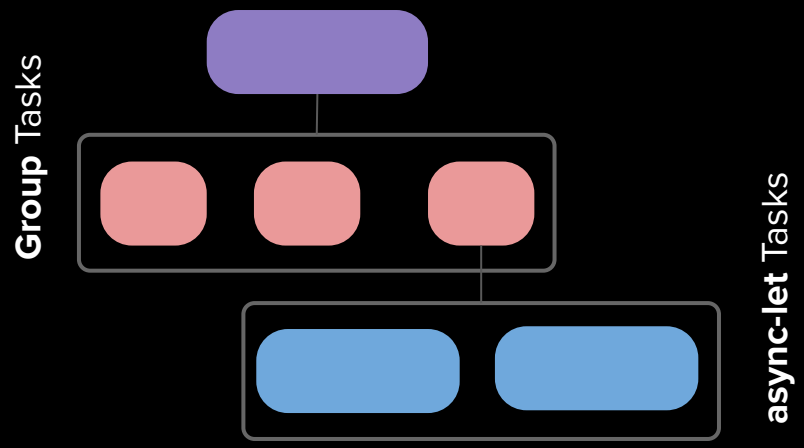


R8s4TqKp9Yw2XuVz3F6G

TaskGroup

avito.tech MTC

```
func fetchThumbnails(for ids: [String]) async throws -> [String: UIImage] {
    var thumbnails: [String: UIImage] = [:]
    try await withThrowingTaskGroup(of: (String, UIImage).self) { group in
        for id in ids {
            group.async {
                return (id, try await fetchOneThumbnail(withID: id))
            }
        }
        for try await (id, thumbnail) in group {
            thumbnails[id] = thumbnail
        }
    }
    return thumbnails
}
```



Actor - серебряная пуля против Data Races?

или нет?

Actors

avito.tech MTC

DispatchQueue.main.async → MainActor

Dispatch Barrier/Locks → Actor

Cooperative Thread pool

Data Race → Actor

Race Conditions

```
class Storage {
    private var data = [ID: Model]()
    private let queue = DispatchQueue(label: "UserStorage.sync")
    func store(_ data: Model) {
        queue.async {
            self.data[data.id] = data
        }
    }
    func loadData(withID id: ID,
                  handler: @escaping (Model?) -> Void) {
        queue.async {
            handler(self.data[id])
        }
    }
}
```

Actors

avito.tech MTC

DispatchQueue.main.async → **MainActor**

Dispatch Barrier/Locks → Actor

Cooperative Thread pool

Data Race → Actor

Race Conditions

```
actor Storage {
    private var data = [ID: Model]()
    func store(_ data: Model) {
        users[data.id] = data
    }
    func model(withID id: ID) -> Model? {
        data[id]
    }
}
```

Actors

avito.tech MTC

DispatchQueue.main.async → **MainActor**

Dispatch Barrier/Locks → **Actor**

Cooperative Thread pool

Data Race → Actor

Race Conditions

```
actor Storage {
    private var data = [ID: Model]()
    func store(_ data: Model) {
        users[data.id] = data
    }
    func model(withID id: ID) -> Model? {
        data[id]
    }
}
```

Actors

avito.tech MTC

DispatchQueue.main.async → **MainActor**

Dispatch Barrier/Locks → **Actor**

Cooperative Thread pool

Data Race → Actor

Race Conditions

```
actor Storage {
    private var data = [ID: Model]()
    func store(_ data: Model) {
        users[data.id] = data
    }
    func model(withID id: ID) -> Model? {
        data[id]
    }
}
```

Actors

avito.tech M T C

DispatchQueue.main.async → **MainActor**

Dispatch Barrier/Locks → **Actor**

Cooperative Thread pool

Data Race → **Actor**

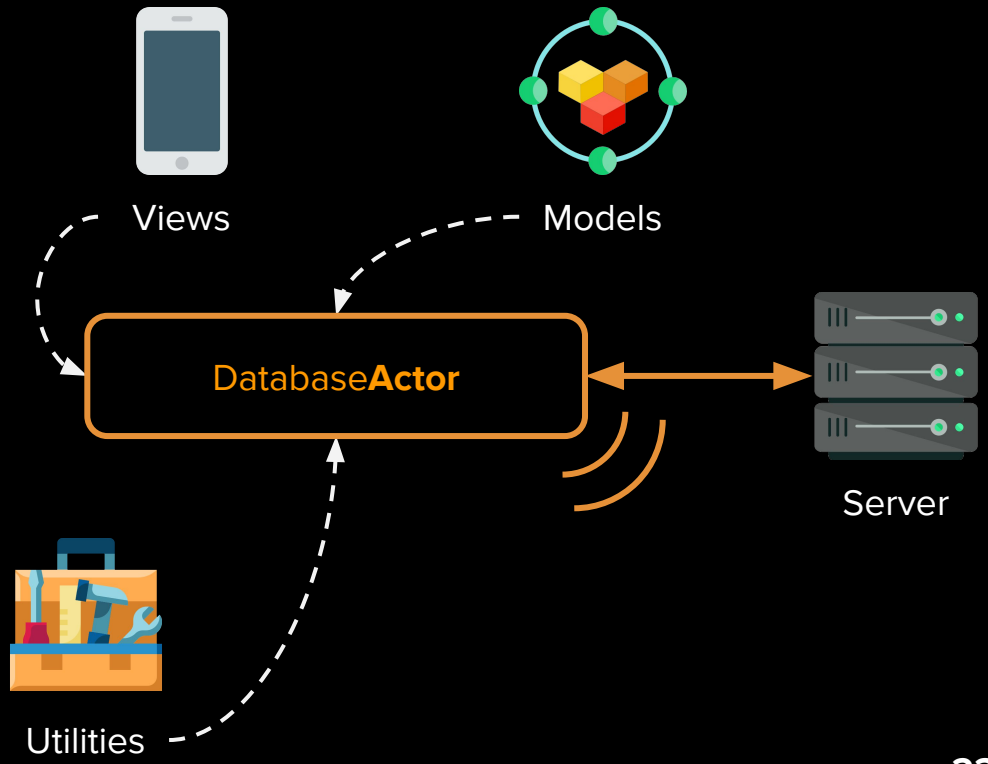
Race Conditions

```
actor Storage {
    private var data = [ID: Model]()
    func store(_ data: Model) {
        users[data.id] = data
    }
    func model(withID id: ID) -> Model? {
        data[id]
    }
}
```

Global Actors | Main Actor

avito.tech MTC

```
@globalActor
struct AvitoActor {
    actor ActorType { }
    static let shared: ActorType = ActorType()
}

@AvitoActor
final class MTS {
    //code
}

@AvitoActor func doSome(_ text: String) {
    //code
}
```



R8s4TqKp9Yw2XuVz3F6G

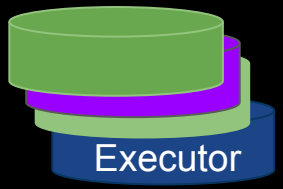
Cooperative Thread Pool

avito.tech MTC

Ограниченное число потоков(CPU)

Особенности

- Рабочие потоки не блокируются
- Избегает чрезмерных переключений контекста



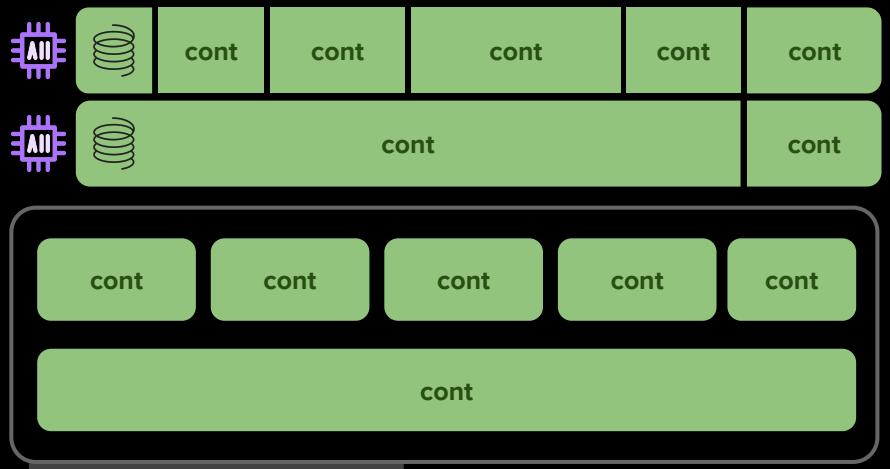
Cooperative Thread Pool

avito.tech MTC

Grand Central Dispatch

Blocked Threads

Swift Concurrency



Blocked Continuations

Executors

avito.tech MTC

Concurrent Executor

```
constexpr size_t dispatchQueueCooperativeFlag = 4;
queue = dispatch_get_global_queue(
            (dispatch_qos_class_t)priority,dispatchQueueCooperativeFlag
);
```

```
JobPriority priority = job->getPriority();
auto queue = getGlobalQueue(priority);
dispatch_async_swift_job(
            queue,
             job,
            (dispatch_qos_class_t)priority,
            DISPATCH_QUEUE_GLOBAL_EXECUTOR
);
```

Serial Executor

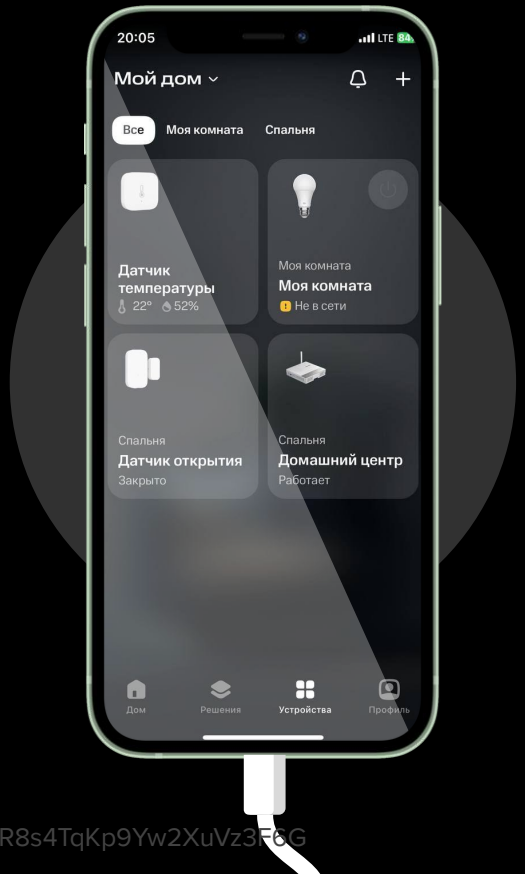
```
class DefaultActorImpl : public HeapObject {
public:
    void initialize();
    void destroy();
    void enqueue(Job *job);
    bool tryAssumeThread(RunningJobInfo runner);
    void giveUpThread(RunningJobInfo runner);
 }

static void setNextJobInQueue(Job *job, JobRef next) {
        *reinterpret_cast<JobRef*>(job->SchedulerPrivate) = next;
}
```

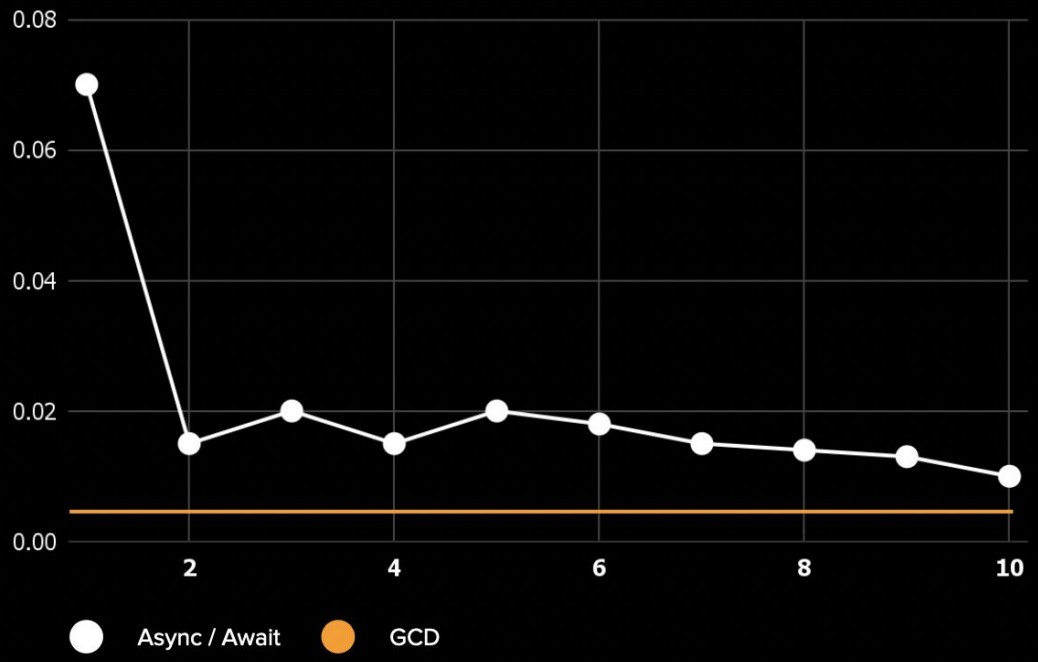
High Up Up Up

III Реальные примеры

Живой пример & метрики

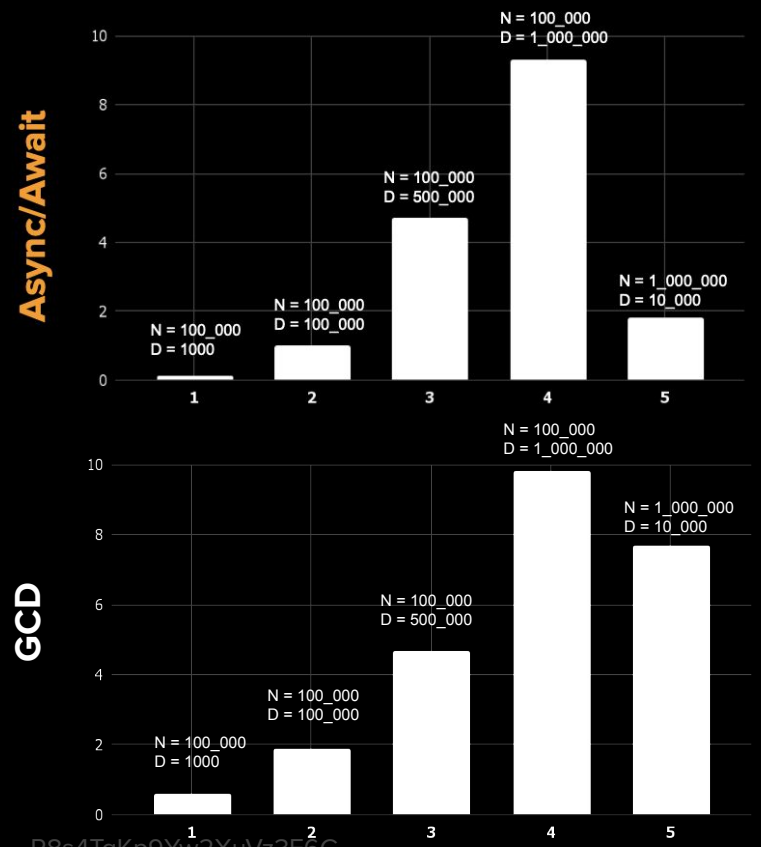
avito.tech MTC

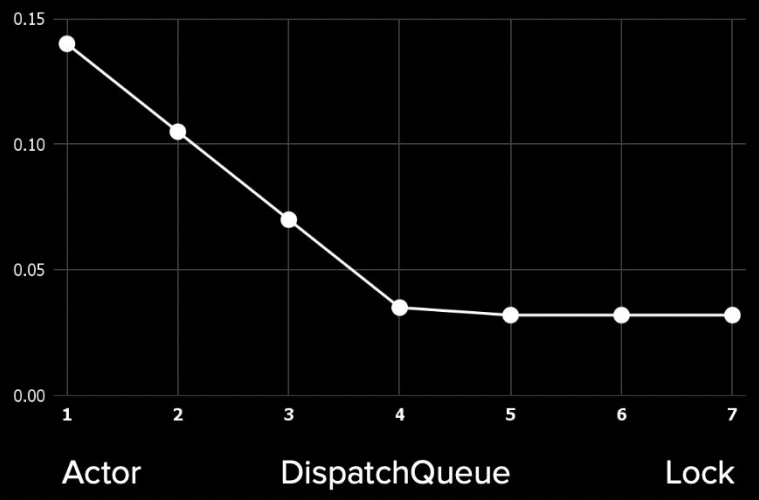




Живой пример & метрики

avito.tech MTC





R8s4TqKp9Yw2XuVz3F6G

Итоги

avito.tech

R8s4TqKp9Yw2XuVz3F6G

async / a wait

async / awai t

asyc / await

asyc / awa it

asyc / awai t

asyc / aw it

async / await

asyc / awa it

asyc / await

avito.tech MTC

Maxim **Surkov**

@surkovmaxim

maxim-surkov

SwifyWay

@SWIFYWAY

R8s4TqKp9Yw2XuVz3F6G

PUBLIC VERSION