

Performance & Scalability Plan

مقدمه ۴/۸/۱ ###

باید در صورت رشد کاربران و افزایش درخواست‌ها، عملکرد `nobatnou.ir` سیستم نوبت‌دهی قابل قبول و در دسترس بودن بالا را حفظ کند. این سند اهداف عملکردی، نقاط ضعف شناخته‌شده، استراتژی‌های مقیاس‌پذیری و برنامه تست بار را تشریح می‌کند.

اهداف سطح سرویس (Service Level Objectives – SLOs) ۴/۸/۲ ###

| هدف | روش اندازه‌گیری SLO | شاخص سطح سرویس (SLI) |

| ----- | ----- | ----- |

| از طریق uptime (ماهانه) | مانیتورینگ ۹۹/۵٪ | ** (Availability) در دسترس بودن ** |
| HTTP checks |

| مانند) APM ثانیه برای ۹۵٪ درخواست‌ها | ابزارهای ۲ < | ** (Latency) زمان پاسخگویی ** |
| Nginx یا لاگ‌های New Relic |

| response برای عملیات نوبت‌گیری | شمارش ۹۹٪ | ** (Success Rate) نرخ موفقیت ** |
| codes 2xx/3xx vs 4xx/5xx |

| پشتیبانی از ۱۰۰ کاربر همزمان بدون افت | ** (Concurrent Users) ظرفیت همزمان ** |
| sessions عملکرد | تست بار و مانیتورینگ |

های فعلی بر اساس زیرساخت فعلی (هاست اشتراکی/سرور مجازی) تعریف شده و با SLO ****توجه**** مقیاس‌گذاری بهبود خواهند یافت.

**** (Bottlenecks) معماری فعلی و نقاط تنگنا ۴/۸/۳ ****

**** معماری فعلی ****

- ****Frontend:**** WordPress سفارشی پلاگین‌های
- ****Backend:**** PHP (Ajax handler با پردازش سنگین در)
- ****دیتابیس:**** MySQL (`nobat_users` و `nobat_system` جدول‌های)
- ذخیره‌سازی: ****فایل‌های وردپرس و آپلودها روی همان سرور ****
- کش پلاگین وردپرس (محدود) (client-side) کش: ****کش مرورگر ****

**** نقاط تنگنا شناسایی شده ****

- | نقطه تنگنا | تأثیر بر عملکرد | راهکار کوتاه‌مدت | راهکار بلندمدت |
- | ----- | ----- | ----- | ----- |
- در بار زیاد | بهینه‌سازی کوئری‌ها، افزودن latency پرس‌و‌جوهای دیتابیس سنگین ****** | افزایش ****** |
- | Read/Write، Replication تفکیک دیتابیس | index |
- فعال‌سازی کش شیء | MySQL و PHP عدم کش در سطح سرور ****** | بار اضافه روی ****** |
- | (Full Page Cache) استفاده از کش کامل صفحه | Redis مانند (Object Cache) |
- PHP-مسدود شدن درخواست‌های همزمان | محدودیت همزمانی در **** Ajax پردازش همزمان **** |
- | (Queue) انتقال پردازش به صف کار | FPM |

(VPS) هاست اشتراکی/منابع محدود** | کاهش عملکرد با رشد کاربران | ارتقاء به سرور مجازی** |
(Cloud) مهاجرت به زیرساخت ابری |

*** **۴/۸/۴ مقیاس پذیری (Scalability Roadmap) برنامه

*** فاز ۱: بهینه سازی (۳-۰ ماه) ***

***:کد -

کاهش تعداد کوئری های دیتابیس در هر درخواست -

برای داده های ثابت (Redis/Memcached) افزودن کش در حافظه -

***:دیتابیس -

(`phone_number`, `position`, `status`). روی ستون های پرکاربرد index ایجاد -

`query_cache` و `innodb_buffer_pool_size` تنظیم پارامترهای -

***:سرور -

مناسب `pm.max_children` با PHP-FPM تنظیم -

opcode cache (OPcache) فعال سازی -

*** (Vertical Scaling) فاز ۲: مقیاس عمودی ***

ارتقاء سرور به ماشین با -

هسته ۴ CPU -

گیگابایت ۸ RAM -

گیگابایت ۱۰۰ SSD -

(Database Server) مهاجرت دیتابیس به سرور جداگانه -

*** (ماه ۱۲-۶) (Horizontal Scaling) فاز ۳: مقیاس افقی ***

PHP. برای توزیع بار بین چند سرور HAProxy یا Nginx لود بالانس: ** استفاده از ** -

***:دیتابیس** -

.(برای خواندن slave برای نوشتن، یک یا چند master یک replication راه اندازی -

`.`created_at`) بر اساس تاریخ `nobat_system` پارتیشن بندی جدول -

و داده های پرتکرار session برای کش Redis cluster کش: ** -

*** (ماه ۱۲+) (Cloud-Native) فاز ۴: معماری مبتنی بر ابر ***

.(AWS, DigitalOcean مانند) انتقال به زیرساخت ابری -

:استفاده از سرویس های مدیریت شده -

برای دیتابیس RDS -

برای کش ElastiCache -

برای استاتیک CloudFront/CDN -

برای ماژول های مستقل (ثبت نام، نوبت دهی، پیامک) microservices طراحی -

*** (Load Testing Plan) برنامه تست بار ۴/۸/۵ ***

****اهداف تست بار ###**

- قابل پشتیبانی (concurrent users) شناسایی حداکثر کاربر همزمان -
- رفتار سیستم تحت بار بیش از حد (breakpoints) تعیین نقاط شکست -
- تأیید بهبودها پس از اعمال بهینه‌سازی -

****ابزارها ###**

- برای شبیه‌سازی کاربران **Apache JMeter** یا **k6** -
- برای مانیتورینگ منابع **Datadog** یا **New Relic** -

****سناریوهای تست بار ###**

**** (Normal Load) سناریو ۱: بار عادی ****

- کاربر همزمان در مدت ۱۰ دقیقه ۵۰ -
- عملیات: مشاهده صفحه اصلی، ورود، مشاهده نوبت -
- زیر ۲ ثانیه، نرخ خطا $1\% > \text{latency}$: انتظار -

**** (Peak Load) سناریو ۲: بار سنگین ****

- کاربر همزمان در مدت ۵ دقیقه ۱۵۰ -
- عملیات: رزرو نوبت همزمان (۲۰ درخواست در ثانیه) -
- زیر ۳ ثانیه، نرخ خطا $5\% > \text{latency}$: انتظار -

**** (Stress Test) سناریو ۳: استرس ****

- کاربر همزمان تا زمان شکست سیستم ۳۰۰ -
- هدف: یافتن حد ظرفیت سیستم -

- **معیارهای اندازه‌گیری** ###
- p95، p99: **زمان پاسخگویی** -
- **RPS) نرخ درخواست در ثانیه** -
- **CPU، RAM، I/O استفاده از** -
- **تعداد اتصالات همزمان به دیتابیس** -

- **فرکانس اجرا** ###
- پس از هر تغییر عمده (ماهانه) -
- قبل از انتشار نسخه جدید -
- (در مرحله بعدی) CI/CD به صورت خودکار با -

Capacity Planning) ظرفیت‌سنجی و پیش‌بینی رشد ۴/۸/۶ ###

- **پارامترهای کلیدی** ####
- نفر (پیش‌بینی سال اول) ۵۰۰: **DAU) کاربران فعال روزانه** -
- HTTP درخواست‌ها در روز: **~ ۱۰,۰۰۰ درخواست** -
- رشد ماهانه: **٪ ۲۰ (پیش‌بینی)** -

****محاسبه منابع مورد نیاز (بر اساس پیش‌بینی ۱۲ ماهه)*****

ماه	کاربران فعال روزانه	درخواست روزانه	سرور پیشنهادی	هزینه ماهانه تخمینی
۱۵ دلار	VPS 2GB	۲,۰۰۰	۱۰۰	۰
۴۰ دلار	VPS 4GB + DB	۱۰,۰۰۰	۵۰۰	۶
کش ۸۰ دلار + ۴GB دو سرور	۳۰,۰۰۰	۱,۵۰۰	۱۲	
زیرساخت ابری (۳ سرور) ۲۰۰ دلار	۱۰۰,۰۰۰	۵,۰۰۰	۱۸	

توجه: **هزینه‌ها بر اساس ارائه‌دهندگان داخلی/خارجی تخمین زده شده است**

**** (Monitoring & Alerting) مانیتورینگ و هشدار ۴/۸/۷*****

****معیارهای مانیتورینگ*****

- CPU > ۸۰٪، RAM > ۹۰٪، Disk I/O > ۱۰۰ms. منابع سرور -
- زمان اجرای کوئری‌های PHP اپلیکیشن: خطاهای -
- (slow queries) دیتابیس: تعداد اتصالات فعال، کوئری‌های کند -

SSL شبکه: ** ترافیک ورودی/خروجی، وضعیت ** -

ابزارها: #####

- **Server:** Netdata, Prometheus + Grafana.
- **Application:** Sentry (برای خطاها), New Relic (برای performance).
- **External Uptime:** UptimeRobot, Pingdom.

هشدارها: #####

- **شدید (Critical):** SLO سرور ۹۵٪ down. (Telegram/Email از طریق)
- **هشدار (Warning):** latency استفاده CPU بالای ۷۰٪، ۳ ثانیه، استفاده (Warning): ** هشدار ** (Email از طریق)

Performance Degradation Recovery) بازایی از افت عملکرد ۴/۸/۸ #####

اقدامات فوری: #####

۱. بررسی لاگها: ** شناسایی درخواست های سنگین یا کند **.
۲. (در صورت قفل) MySQL، PHP-FPM: ** ری استارت سرویس ها **.
۳. پاک سازی کش: ** کش دیتابیس یا کش صفحات **.
۴. overload موقت برای جلوگیری از rate limit محدود سازی ترافیک: ** کاهش **.

****اقدامات بلندمدت پس از حادثه**** #####

(Root Cause Analysis – RCA) تحلیل ریشه‌ای -

runbooks به‌روزرسانی مستندات و -

اعمال تنظیمات پیشگیرانه -

****آینده (Auto-scaling Plan – برنامه مقیاس خودکار ۴/۸/۹**** #####

پیش‌نیاز: **مهاجرت به زیرساخت ابری**

****Instances (ماشین‌های مجازی****

PHP (Frontend) سرورهای **: (Group) گروه** -

به مدت ۵ دقیقه $CPU > 70\%$:معیار مقیاس** -

instance حداکثر: **۵** -

****دیتابیس****

auto-scaling با قابلیت (مانند AWS RDS) استفاده از سرویس مدیریت‌شده -
storage/IOPS.

Scalability Costs) هزینه‌های مقیاس‌پذیری ۴/۸/۱۰ ***

مرحله	اقدامات	هزینه ماهانه تخمینی	تأثیر بر عملکرد
	-----	-----	-----
تنظیمات	۱۰۰~ هزار تومان (نیروی انسانی)	بهبود ۳۰-۵۰٪ index بهینه‌سازی**	کش، **
قوی‌تر	۴۰-۶۰ دلار	بهبود ۱۰۰-۲۰۰٪ VPS مقیاس عمودی**	ارتقاء سرور به**
مقیاس افقی**	افزودن سرور و لود بالانس	۱۰۰-۲۰۰ دلار	بهبود ۳۰۰-۵۰۰٪**
ابر**	مهاجرت به زیرساخت ابری	۲۰۰-۵۰۰ دلار	بهبود ۱۰۰۰٪ + قابلیت اطمینان بالا**

توجه:** هزینه نیروی انسانی برای پیکربندی و نگهداری جداگانه محاسبه شده است**

نتیجه‌گیری ۴/۸/۱۱ ***

با معماری فعلی می‌تواند تا ۱۰۰** کاربر همزمان** را با عملکرد قابل `nobatnou.ir` سیستم قبول پشتیبانی کند. با اجرای برنامه بهینه‌سازی و مقیاس‌پذیری مرحله‌ای، سیستم قادر خواهد بود تا های تعریف‌شده مدیریت کند. کلید موفقیت، **مانیتورینگ SLO ۵۰۰** کاربر همزمان** را با حفظ مستمر، تست بار دوره‌ای و انعطاف‌پذیری در انتخاب زیرساخت** است. این برنامه به‌گونه‌ای طراحی شده که هزینه‌ها با رشد درآمد کسب‌وکار افزایش یابد.