Ответы на вопросы по лекции "Аппаратное обеспечение и развертывание серверов баз данных"

# 1. Компромиссы при выборе "железа" (CPU и диски)

CPU:  
- Для OLTP важна высокая частота ядра, чтобы каждая транзакция выполнялась максимально быстро.  
- Для OLAP важнее количество ядер, так как аналитические запросы распараллеливаются.  
- В гибридной нагрузке приходится искать баланс: взять процессор с достаточной частотой, но и с большим числом ядер.  
  
Хранилище:  
- Для OLTP решающим является IOPS (скорость множества мелких операций).  
- Для OLAP критична пропускная способность (Throughput).  
- Компромисс — NVMe SSD.  
  
Разделение данных по дискам:  
- Данные → быстрый массив SSD/NVMe.  
- WAL → отдельный сверхбыстрый NVMe.  
- Temp files → отдельный SSD.  
Такое разделение снижает конкуренцию за I/O.

# 2. Расчет оперативной памяти и риски при нехватке

Формула: Total RAM = Working Set Size + OS Overhead + Peak Load Buffer.  
Если рабочий набор вырос на 25%, но память не увеличена:  
- Буферный кэш станет меньше, чем рабочий набор → больше промахов кэша → рост задержек.  
- Недостаток памяти для сортировок приведет к использованию диска для temp files → резкое замедление.  
  
Проявления:  
- Увеличение задержек запросов.  
- Рост обращений к temp files.  
- Перегрузка подсистемы хранения при пиках.

# 3. Архитектура хранения и отказоустойчивость (ZFS vs. RAID)

Аппаратный RAID на NVMe может стать бутылочным горлышком (контроллер ограничивает IOPS).  
  
Преимущества ZFS:  
- Использует полные возможности NVMe.  
- Встроенные снапшоты, дедупликация, сжатие.  
- Контроль целостности данных.  
- Гибкость добавления дисков и изменения конфигурации.  
  
Итого: ZFS лучше для высокопроизводительных СУБД.

# 4. Pets vs. Cattle и аварийное восстановление

Pets: сервер уникален, чинят вручную → долгие RTO.  
Cattle: серверы заменяемые, восстанавливаются автоматически через IaC.  
  
Инструменты: Terraform, Ansible, Kubernetes.  
Преимущество: мгновенная замена узлов, масштабирование без простоев.

# 5. Безопасность и производительность (TDE vs. LUKS)

TDE:  
- Шифруются таблицы.  
- Влияет на CPU базы.  
  
LUKS:  
- Шифруется весь диск.  
- Проще, меньше нагрузка на СУБД.  
  
Сценарии:  
- При высокой нагрузке на запись лучше LUKS.  
- Если нужна защита отдельных данных — TDE.

# 6. RPO и RTO как бизнес-требования

RPO = допустимая потеря данных.  
RTO = допустимое время простоя.  
  
Примеры:  
- Банк: RPO ≈ 0, RTO < 5 мин.  
- Стриминг: RPO — часы, RTO — 1–2 часа.

# 7. Проактивное vs. реактивное обслуживание

Если не делать автовакуум и не перестраивать индексы:  
- Таблицы разрастаются мертвыми строками.  
- Индексы фрагментируются.  
  
Метрики:  
- Рост размера таблиц.  
- Увеличение времени запросов.  
- Рост блокировок.

# 8. Вертикальное масштабирование без простоя

Использовать балансировщики и пулы соединений (PgBouncer, HAProxy).  
- Старый сервер можно перевести в read-only.  
- Новый нагоняет репликацию.  
- Переключение соединений без явного простоя для пользователей.

# 9. Производительность закладывается на уровне железа

Опровержение «купим потом»:  
- Диски: медленные SATA не вытянут нагрузку, потом миграция сложна.  
- Память: если не хватает RAM, база будет постоянно лезть на диск.  
- Сеть: 1GbE ограничивает репликацию, апгрейд потребует замены инфраструктуры.  
Вывод: фундаментальные ошибки нельзя компенсировать позже.

# 10. Конфигурация сервера для стартапа

Начало:  
- CPU среднего уровня (EPYC/Xeon Silver).  
- RAM 64–128 ГБ.  
- NVMe под данные, отдельный диск под WAL.  
- Сеть 10GbE.  
  
Стратегия:  
- IaC для автоматизации.  
- «Cattle»-подход.  
- Возможность вертикального и горизонтального масштабирования.  
Итог: бюджетный старт, плавный рост.