TOP1

主办方 msup

技术立身，融合创新

----一个跨界架构师程序员的AI时代生存之道

周恩：华为技术专家兼软件教练，原京东交易架构师兼极客导师

本演讲仅代表个人观点，与任何公司无关！

讲师简介

周恩

华为技术专家兼软件教练
原京东交易平台架构师兼“京东GEEK发现代码之美”评审导师

2019、2020年连续两届华为全球技术服务部代码前20强；设计实现过京东“我的关注”（QPS XXX万，日最大访问量百亿次级）、“Plus会员”等多个核心系统；是多个软件发明专利和著作权（含开发工具和商业软件）的唯一作者，修改过5+个开源软件。对产业链感兴趣。

(微信：isharedata)

摘要

未来是数据驱动的商业智能！AI的影响是全面、深刻且不可逆转的，受限于投资、算法和数据的高门槛，以及硬件的可获得性风险，多数人和组织必然只是AI使用者角色，AI也将减少普通从业人员岗位（含软件），但AI尚不足以吞噬一切。

演讲者仅代表个人，分享在这个内卷又充满AI挑战的时代，先生存后发展的个人应对之道，坚持技术立身，走融合创新之路，包括追求技术相对**精深**，**融合**技术、业务和管理做复合型人才，用好**新兴技术**，**使用AI**技术帮助工作学习。演讲者会结合实践案例来分享，包括“我的关注”融合多种技术来实现服务器降低10倍反而性能更高更稳定、企业管理软件演进过程中是如何体现融合创新和竞争优势的，新兴高效能技术是如何消减AI影响力的。演讲者也会提醒你注意AI时代的平衡相对论，并在内卷时代注重内生强大。

希望我们不变初心，终身学习，使能商业成功，成为技术、业务和管理集大成的AI时代弄潮儿。

目录

纲要：

- 关键判断：弱人工智能阶段
- 技术精深：朝金字塔顶端走
- 融合创新：走复合型创新路
- 新兴技术：坚持吃技术红利
- 内在强大：做事不卷求幸福

案例：

- “我的关注”融合多种技术来实现服务器降低10倍反而性能更高更稳定
- 企业管理软件演进过程：从产品到产品组合到产业链，从关系型数据到大数据及AI
- Doris和云原生技术

AI现状与影响

- AI已学习人类所有已数据化公开信息，包括部分商业信息
- AI已学习所有Github及网上代码，包括不友好授权协议代码
- AI已可常态化辅助人类，包括编程、多媒体、设计、文案，等等

AI，代表有挑战的未来，人类必须接受和使用她，善用者能

AI，已经具备成为超人的基础，迭代正在加速，已到临界点，已有意识萌芽

AI，风险也必加剧，伦理风险在被忽略，对抗AI也有必要性

当下我对AI的关键判断

- 是从大规模数据中归纳出知识，**数据**和**算法**是基础
- 是人类全部问题中的问题子集，是**有限小范围子集**
- 需要大规模**算力**基础设施支撑，**大投资**可得是关键

算力基础设施，以及关联**基础软件**，是需要突破的根基基础技术

AI，当下暂时处于弱人工智能级阶，暂时不具备意识（已有萌芽），已可减少简单可重复性工作

结合大数据和常规软件能力，**使用者创新**，AI使能商业成功，是技术生存之本

AI基础的超高门槛，决定了多数人应该是AI创新使用者

- 投资：以亿美元为投资单位
- 算法：底层核心算法门槛高
- 数据：少数公司能用好数据
- 政策：连续可获得性高风险

极少数人是AI创造者，**多数人应是AI使用者，聚焦于使用者创新**

中国及其伙伴的巨大市场可以培育群体自主创新，向所有根技术创新者致敬

数据及算法影响AI成熟度，技术人员还有一定生存空间

- 数据：多数数据还暂时未被AI所使用，或者暂未数字化
- 算法：算法还不能汇集全部数据计算，并达到人类智能，或者成本过高

数据和算法限制，决定了**当前AI能力多数尚在中人之上，部分能力已远超人类**

技术精深，是广度基础上求深度，在质效上减弱AI能力影响

- 聚焦质效，优先**以用户视角看技术深度**
- 兼顾广度，才能够正确评估AI结果优劣
- 谨慎平台，不要把平台能力当自己能力
- 须借力AI，获得更多灵感并加深技术力

算法和具体实现深度是有争议的，**个人倾向于问题驱动**，充分借力AI增强能力
除少数创造者外，多数人应该聚焦于应用优先，避免在中间层次消磨过多精力

技术精深，深与浅是辩证的，需特别注意管理者因素

- 大道至简，是境界，体现技术深度和利他
- 管理竞争，要看人，可能会把至简当无能

老板和员工心态，未必真统一，团队中利益、格局和见识会加重分歧，你务必自行判断自己担责

我遵从老板心态，信奉技术干净原则，也必愿意承担后果

技术精深，平台是双刃剑，不一定是技术深，容易滋生屎山和贼船

- 架构腐化，常态缺乏架构设计和看护
- 低质编码，炫技、垃圾遗留、缺注释
- 缺乏前瞻，对业务技术缺乏长远规划

屎山通常与自负的“大牛”和不负责的“撞钟的和尚”有关

平台，可以提高AI介入门槛，也容易固化并降低使用者技能

技术精深，竞争要回归本质初心，自研框架平台不是唯一

- 回归本质，如，数据容量、性能、可靠性
- 商业价值，**商业体现价值，技术是您支撑**
- 提升格局，不要狭隘的增加壁垒，给发展挖坑

商业短视，技术易被商业抛弃，最终商业发展受限的情况很常见，希望这个有改变

示例（仅示例性演示）：JPA标准技术极简高效，也支持原生SQL高性能

```java
@Repository
public interface OrderRepository extends CrudRepository<Order, Long> {

    @Query(value =
            """
            select a.province, a.city, a.county, to_char(o.create_date, 'YYYY') as orderYear, to_char(o.create_date, 'YYYY-MM') as orderMonth,
                count(distinct(o.order_id)) as orderCounts, sum(oe.quantity) as orderQuantities, sum(oe.price * oe.discount * oe.quantity - oe.deduction) as totalAmounts
            from ssb_order o
            join ssb_order_item oe on o.order_id = oe.order_id
            join ssb_address a on o.customer_id = a.customer_id
            where o.order_time between ?1 and ?2
            group by a.province, a.city, a.county, orderYear, orderMonth
            order by a.province, a.city, a.county, orderYear, orderMonth, totalAmounts desc
            """,
            countQuery =
            """
            select count(*) from (
                select a.province, a.city, a.county, to_char(o.create_date, 'YYYY') as orderYear, to_char(o.create_date, 'YYYY-MM') as orderMonth,
                    count(distinct(o.order_id)) as orderCounts, sum(oe.quantity) as orderQuantities, sum(oe.price * oe.discount * oe.quantity - oe.deduction) as totalAmounts
                from ssb_order o
                join ssb_order_item oe on o.order_id = oe.order_id
                join ssb_address a on o.customer_id = a.customer_id
                where o.order_time between ?1 and ?2
                group by a.province, a.city, a.county, orderYear, orderMonth) as t
            """,
            nativeQuery = true)
    Page<TopAreaOrderProjection> findTopAreaOrder(LocalDateTime orderTimeStart, LocalDateTime orderTimeEnd, Pageable pageable);

    Collection<Order> findByOrderTimeBetween(LocalDateTime orderTimeStart, LocalDateTime orderTimeEnd);
```

```java
public interface CrudRepository<T, ID> extends Repository<T, ID> {
    1 implementation
    <S extends T> S save(S entity);

    no usages  2 implementations
    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    2 usages  1 implementation
    Optional<T> findById(ID id);

    no usages  1 implementation
    boolean existsById(ID id);

    2 implementations
    Iterable<T> findAll();

    no usages  2 implementations
    Iterable<T> findAllById(Iterable<ID> ids);

    1 implementation
    long count();

    1 usage  1 implementation
    void deleteById(ID id);

    1 implementation
    void delete(T entity);

    no usages  1 implementation
    void deleteAllById(Iterable<? extends ID> ids);

    no usages  1 implementation
    void deleteAll(Iterable<? extends T> entities);

    no usages  1 implementation
    void deleteAll();
}
```

深一点，JPA支持原生SQL（多种）；更深一点，投影技术强化复杂场景代码效率

红线处可以使用虚拟计算列技术（有限制也有数据库差异）。虚拟计算列可简化计算并减少数据不一致

融合创新，本质是以系统性和人因因素，借力AI能力来减弱AI能力影响

- 技术融合 --> 技术+业务 --> 技术+业务+管理
- 人因因素，本质是人类意识的独特性增强，定量科学加艺术不确定性，AI优势就可能越低

业务体现价值，技术是您支撑，管理起正向（或负向）作用

融合的本质，也是利用系统性增强复杂性和不确定性，弥补AI对复杂关联问题能力不足，并结果简化

技术融合，应以架构设计实现能力为基础，并借力AI做增强

- 从单技术实现，转为关注整体设计实现
- 从单技术最优，转为关注系统整体最优
- 单技术实现上，须借力AI获取能力灵感

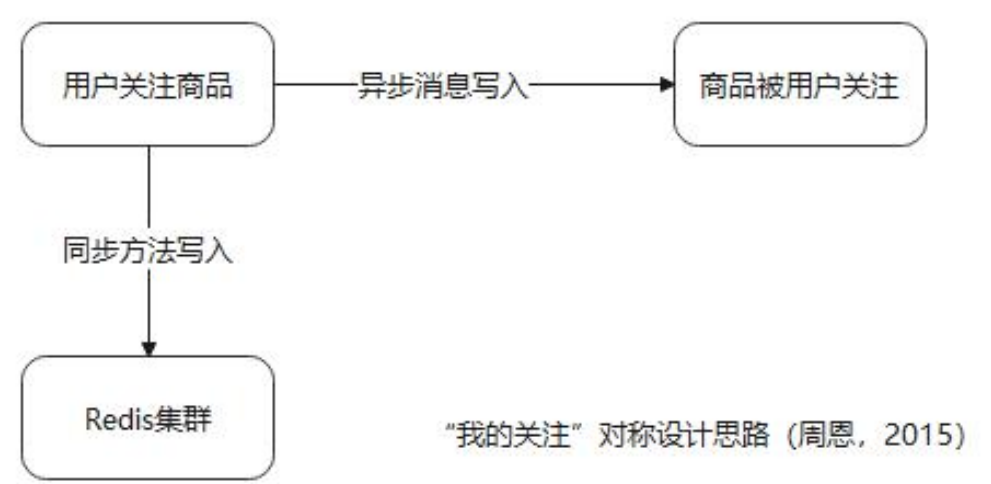
架构设计，应以深厚技术为基础，避免做平台依附型伪架构师

关注整体，又不排斥在单一技术实现上，充分利用AI能力来做增强，是更合理的实践

案例：我的关注，融合多种技术，服务器降低10倍反而性能更高更稳定

- 问题：用户关注商品、店铺等，数据读写（尤其读）性能不足，大促时容易挂掉

- 解决方案：
  - ✓ 简化技术：去掉HBase、Solr/Lucene、LevelDB等，仅用MySQL+Redis+消息
  - ✓ 优化设计：对称数据库表设计，异步消息，分库分表扩容
  - ✓ 数据服务：独立数据访问服务，减少数据库连接资源争抢
  - ✓ 其他优化：缓存扩容、缓存部署模式和位置、均衡算法等



“我的关注”对称设计思路（周恩，2015）

注：本案例适合单独完整分享，这里仅简单介绍

技术与业务融合，就是用技术增强业务，实现双轮驱动的竞争优势

- 业务：从需求到产品，从产品到产品组合，再到产业链
- 技术：从准确实现需求，到技术可演进，并满足业务可演进

技术在于内深外简**可控的灵活性**，业务在于构建可演进的**商业护城河**

技术业务两相结合，实现商业竞争优势能可持续演进，是以复杂度加人因创造性减弱AI能力影响

技术、业务和管理融合，就是继续增强人因决策因素

- 管理：**从战略到执行，是有序和可控**，是人因因素之美
- 业务：从需求到产品，从产品到产品组合，再到产业链
- 技术：从准确实现需求，到技术可演进，并满足业务可演进

管理是实现**战略到执行的有序可控**，业务在于构建可演进的**商业护城河，**技术在于内深外简**可控的灵活性**
技术业务管理相结合，在可预见的未来，都是可以减弱AI能力影响的

案例：企业管理软件，在业务和技术之上，都会有管理的决策因素

- 业务：采购 --> 生产 --> 销售 --> 服务
- 技术：交易记录 --> 大数据 --> 数据分析与AI增强
- 管理：业务最终设计、技术最终设计

以小渐大，从产品到产品组合，再到产业链，本质上就是要**构建企业商业护城河**

未来是数据驱动的商业智能，数据和智能必然成为企业商业技术底座，技术是驱动轮

业务和技术的发散和收敛，均是以人因管理为决策的，是技术与艺术的结合

新兴技术，就是顺应潮流，吃新兴技术红利

- **AI、大数据、云原生**等技术快速发展
- 对数据中台、微服务的检讨也在继续
- 可演进的大数据叠加AI的架构是关键
- AI正在快速迭代、**简化和易用**是关键

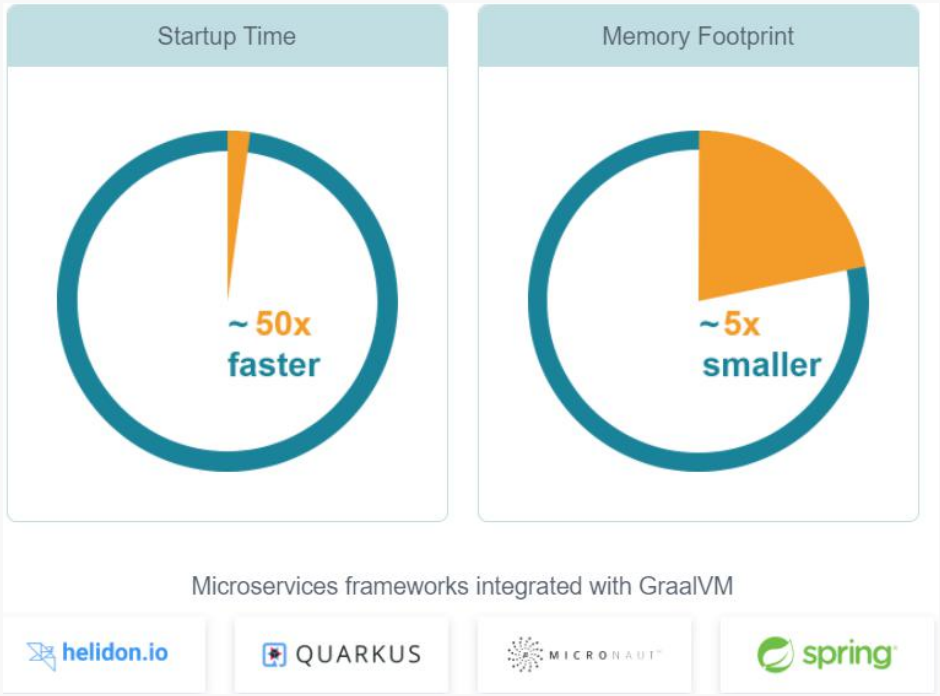
以Hadoop为代表的早期技术还会演进，ClickHouse等触发了更多大数据新技术，如Doris等
结合GraalVM和Java语言新特性，Quarkus等触发了Spring Boot 3等更多云原生新技术

案例：现代化的Doris，数据读写性能大幅提升，并解决数据更新问题

- Doris对比ElasticSearch等，读写性能有大幅度提升，对比MySQL等传统数据库提升可达数百倍
- Doris对比ClickHouse，解决了数据更新问题（注：ClickHouse有多种手段实现不优雅数据更新效果）

致谢：本页图片由Apache Doris官方团队友情提供

案例：云原生性能优势明显



- 云原生助力启动速度提升约50倍，内存消耗减少约5倍
- 结合Jakarta EE 9/10规范和Java语言新特性，可以大幅提升研发效率

致谢：本页图片分别来自https://www.graalvm.org和https://quarkus.io

其他，要关注技术管理对个人成长的影响

- 团队组织：高低搭配，重视遴选，尊重人性
- 人才培养：**基础扎实、愿意上进、懂得感恩**
- 有效协作：团队协作、避免无效需求和返工
- 软件工程：过程要适配裁剪，敏捷也要设计

与优秀的人合作会让自己更优秀！**合作者厉害，意味着自己更厉害**

避免KPI伪需求、避免没想清楚就匆忙编码、避免高强度拼体力造成团队崩溃

案例：稳定上进的团队管理要考虑人性

招聘部门40%员工约33个月内仅1人离职，人少2K也愿来，培养出多个专家和经理

如何做到：

- ✓ **价值观：愉快、成长、挑战、成就感、最好钞票也合适**
- ✓ 领导力：**领导不是领导岗位**，架构师也可以发挥领导力
- ✓ 内省力：以己及人，人性考量的同理心，方可用人以心

管理的初阶是管控，中阶是管控结合人性，最高境界是用人以心

人性化管理需要组织支持，也对管理有更高要求，要专业和责任，搞得不好就会是放羊

要追求质量和效率，看得长远，就不要拼时间搞内卷，除非真的只是搬砖

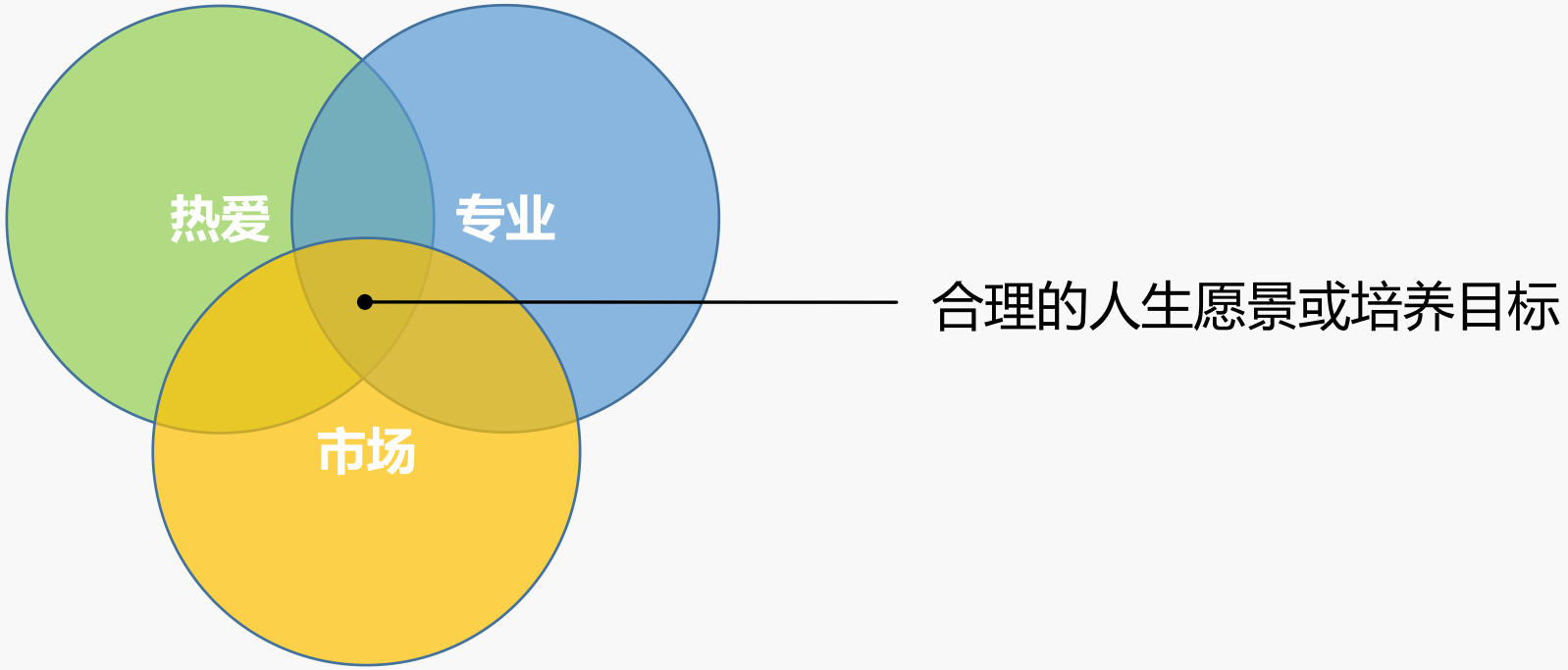
其他，内心强大，追求幸福

- **避免被物化**，留点时间给自己，学习、家人、锻炼
- 潜居抱道，以待其时，有机会就**不折腾**的做点事情
- 如果没有机会，争取做个**幸福**的平凡人

随时做好准备，但不是说有准备就一定会有成功，要合理预期

良好心态和认识自我，可以帮助我们在内卷时代相对平和幸福

不管在哪个时代，都需要坚持学习的成功三环



市场、热爱和专业，基本确定了一个人的高度

热爱和坚持，可以把平庸变为专业；专业和市场，可以把无感变为热爱

内容回顾：谢谢您的聆听，期待与您再相见！

- 关键假设：AI已具备超人基础，弱人工智能阶段，聚焦使用者创新
- 技术精深：广度基础上求深度，深与浅是相对的，要避免平台依附
- 融合创新：技术、业务、管理，融合创新，提升价值减弱AI影响力
- 新兴技术：AI、大数据和云原生发展较快，对技术检讨也必将持续
- 内在强大：管理影响团队，爱他人爱自己，有机会做事，平和幸福

希望认识更多新朋友，融合，创新

(微信：isharedata)

分享赢得更多，希望认识更多格局、专业、友好、还有点想法的新朋友

TOP100®

主办方 msup®

微信官方公众号：壹佰案例
关注查看更多年度实践案例