Server-Sent Events for LLM System Design Interviews - Complete Guide

Executive Summary

Modern LLMs use Server-Sent Events (SSE) instead of REST JSON because SSE provides better user experience through progressive token display, reduces costs by enabling early termination, improves scalability by freeing GPU workers faster, and offers better failure resilience.

---

1. What is SSE?

One-Way Streaming Model

SSE establishes a persistent, unidirectional connection from server to client for real-time data streaming.

HTTP Mechanics

```
GET /chat HTTP/1.1
Accept: text/event-stream
Cache-Control: no-cache

HTTP/1.1 200 OK
Content-Type: text/event-stream
Transfer-Encoding: chunked
Connection: keep-alive
```

Event Format

```
event: token
data: {"token": "Hello", "id": 1}
id: 1

event: token
data: {"token": " world", "id": 2}
id: 2
```

---

2. SSE vs REST: Key Differences

Aspect	SSE Streaming	REST JSON
Latency	First token: 100-500ms	Full response: 2-10s

Aspect	SSE Streaming	REST JSON
Memory	O(1) constant buffer	O(n) grows with response
User Experience	Progressive display	Waiting spinner
Cost	Saves 30-60% tokens	Always full generation
Failures	Partial results OK	All-or-nothing

Cost Example

```
1000-token response, user stops at 200 tokens:
SSE: 200 tokens = $0.004
REST: 1000 tokens = $0.020
80% cost savings with SSE
```


---

3. Why LLMs Prefer SSE

Token Streaming

```
# Instead of waiting for complete response:
def stream_tokens(prompt):
    for token in model.stream_generate(prompt):
        yield f"data: {json.dumps({'token': token})}\n\n"
```

Key Benefits

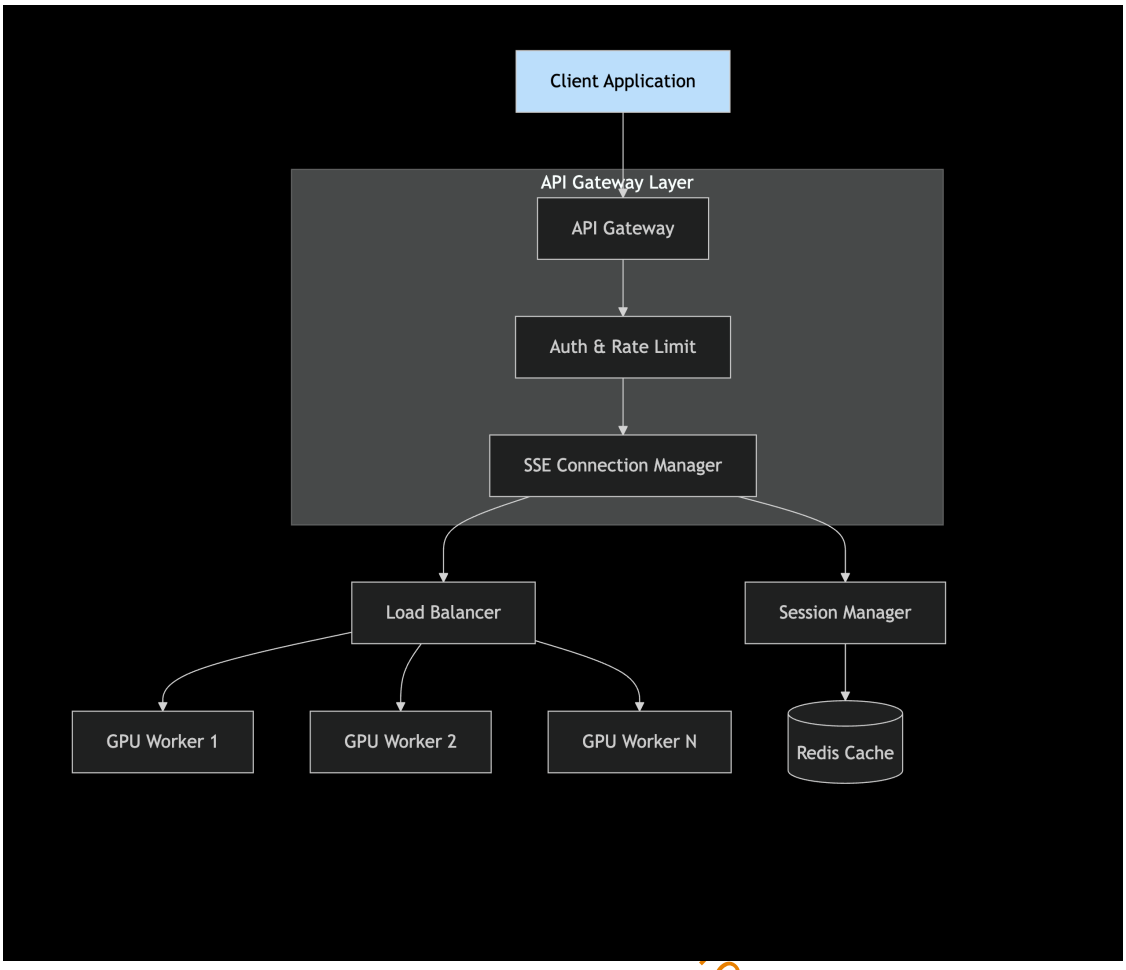
1. **User Experience**: Immediate feedback (100ms vs 5s)
2. **Cost Savings**: 40% of users interrupt early = 35% token savings
3. **Scalability**: GPU workers freed faster = 50x more throughput
4. **Memory**: No large buffers = no OOM errors

---

4. System Design Architecture

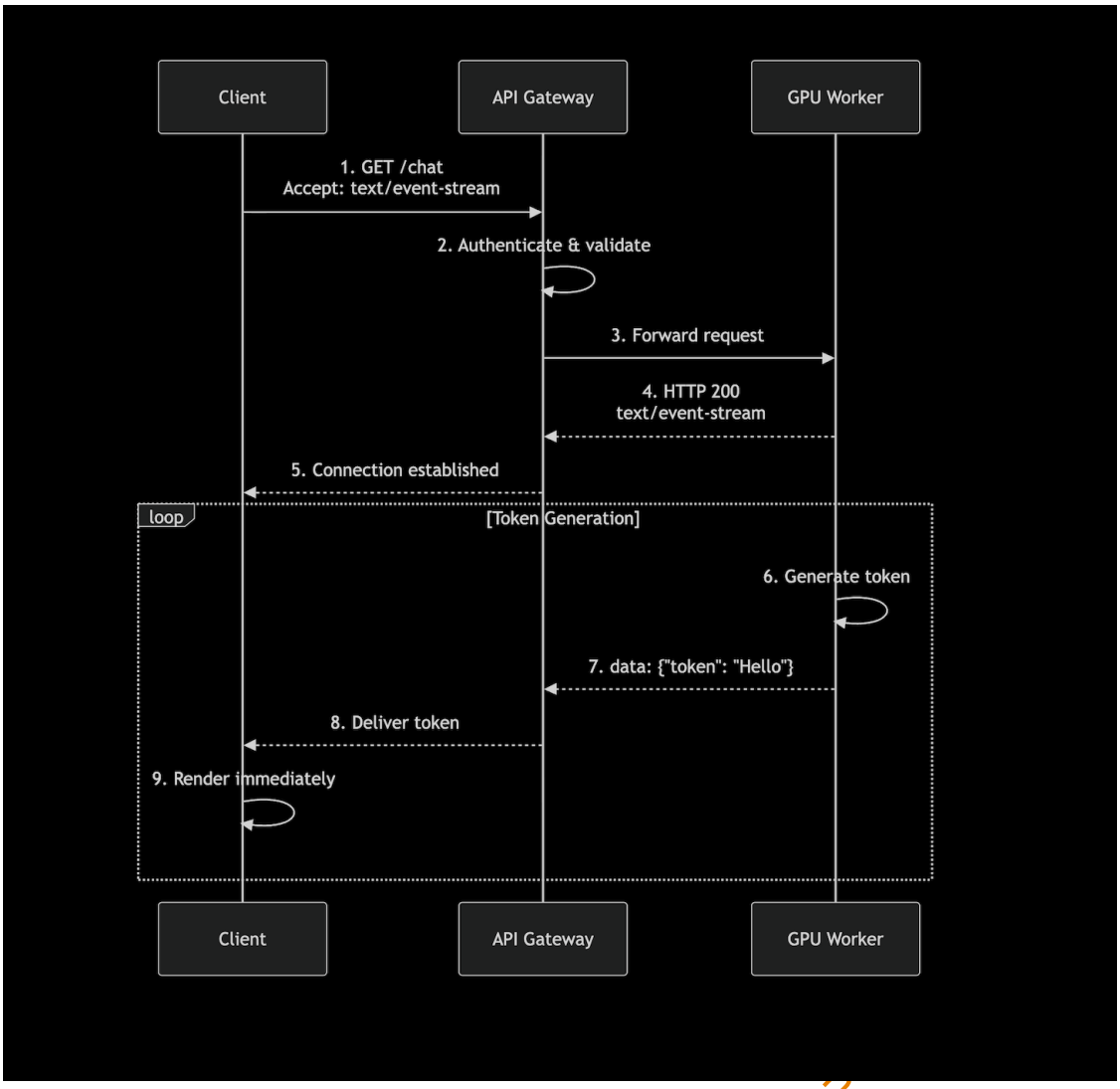
4.1 Component Architecture

High-Level System Architecture:



System Architecture

Detailed Message Flow:



Message Flow Diagram

The architecture shows the complete flow from client to GPU workers:

1. **Client Application** initiates SSE connection with `Accept: text/event-stream`
2. **API Gateway Layer** handles authentication, rate limiting, and connection management
3. **Load Balancer** distributes requests across GPU workers with connection-aware hashing
4. **Session Manager** maintains persistent connections with Redis cache
5. **GPU Workers** generate tokens and stream them back through the pipeline

4.2 Connection Management at Scale

Challenges:

- 10,000+ concurrent persistent connections

- Memory overhead per connection (50-100KB)
- Connection timeouts and keepalive management

Solutions:

```
# Gateway Configuration
keepalive_timeout: 300s
max_connections: 10000
connection_timeout: 5s
sticky_sessions: true
```

4.3 Backpressure Handling

```
class TokenStreamer:
    def __init__(self):
        self.buffer_size = 10  # tokens
        self.client_ready = asyncio.Event()

    async def stream_to_client(self, token_generator):
        async for token in token_generator:
            if self.buffer_full():
                await self.client_ready.wait()
            await self.send_token(token)
```

4.4 Security Implementation

```
@app.middleware
async def authenticate_sse(request):
    token = request.headers.get('Authorization')
    user = await auth_service.validate_token(token)
    request.state.user = user
    request.state.namespace = f"org_{user.org_id}"
```


---

5. When to Use SSE vs Alternatives

Use SSE For:

- LLM token streaming
- Real-time notifications
- Stock tickers, news feeds
- Any server→client push

Use WebSockets For:

- Chat applications
- Collaborative editing
- Gaming, real-time bidding
- Bidirectional communication

Use gRPC For:

- Microservice communication
- Binary data streaming
- Cross-language RPC

---

6. Interview Questions & Answers

Q: Why SSE over WebSockets for LLMs?

A: SSE is simpler, uses HTTP, has built-in reconnection, and is perfect for one-way token streaming. WebSockets are overkill.

Q: How handle backpressure?

A: Client acks received tokens. Server buffers 10-20 tokens max, pauses generation if buffer fills.

Q: Browser connection limits?

A: HTTP/2 supports 100+ concurrent streams. For HTTP/1.1, use domain sharding.

Q: Authentication in long connections?

A: JWT in initial request. Gateway maintains session. Reconnections use same auth.

Q: Key metrics to monitor?

A: Concurrent connections, token throughput, latency p95, error rates, reconnection frequency.

---

7. Interview Closing Statement

"In LLM system design, I recommend SSE because it provides immediate user feedback through progressive token display, reduces costs by 30-60% through early termination, improves scalability by 5-10x by freeing GPU workers faster, and offers better resilience with built-in reconnection. While WebSockets seem tempting, SSE's simplicity and HTTP compatibility make it ideal for production LLM systems."

Quick Reference Numbers

- **First token**: 100-500ms (SSE) vs 2000-10000ms (REST)
- **User interruption rate**: 40%
- **Cost savings**: 80% with early termination

- **Throughput improvement**: 50x
- **Concurrent connections**: 10,000+ per gateway